

PEARSON

ALWAYS LEARNING

100 SOA Questions

Asked and Answered

KERRIE HOLLEY
DR. ALI ARSANJANI

Custom Edition for IBM

Taken from:
100 SOA Questions: Asked and Answered
by Kerrie Holley and Dr. Ali Arsanjani

Content Excerpted from
Pearson Technology Group by IBM.
Not for Resale or Distribution.

Taken from:

100 SOA Questions: Asked and Answered
by Kerrie Holley and Dr. Ali Arsanjani
Copyright © 2011 Pearson Education, Inc.
Published by Prentice Hall
Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

This special edition published in cooperation with Pearson Learning Solutions.

All trademarks, service marks, registered trademarks, and registered service marks are the property of their respective owners and are used herein for identification purposes only.

Pearson Learning Solutions, 501 Boylston Street, Suite 900,
Boston, MA 02116
A Pearson Education Company
www.pearsoned.com

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 XXXX 16 15 14 13 12 11

000200010270730547

RG

PEARSON

ISBN 10: 1-256-05392-9
ISBN 13: 978-1-256-05392-7

Contents

	Introduction	1
	About This Book	2
	Intended Audience	2
	How This Book Is Organized	3
Chapter 1	SOA Basics	5
	SOA Basics: Q&A	5
	1. What Is SOA?	5
	2. Is SOA an Architectural Style?	7
	3. What Are the Fundamental Constructs (the DNA) of SOA?	9
	4. What Is the Difference Between a Web Service and an SOA Service?	14
	5. What Makes a Project an SOA Implementation?	15
	SOA Basics: Key Concepts	16
Chapter 2	Organization	19
	Organization: Q&A	20
	6. How Does Business/IT Alignment Change Because of SOA?	20
	7. Which Joint Business/IT Processes Change Because of SOA?	23
	8. What Organization Structures Should Be Established for SOA?	24
	9. What Is the Role of Organizational Change Management to SOA?	30
	10. How Can Organizational Barriers to SOA Success be Removed?	32
	11. How Should Organizations Address Funding for Services?	33
	12. How Should Organizations Address Prioritization for Shared Services?	37

	13. What Are Service Owners?	38
	14. What is the Value of Classifying Services?	39
	15. Who Owns Service Reuse?	40
	16. What Are the Common Organizational Pitfalls When Adopting SOA?	41
	Organization: Key Concepts	42
Chapter 3	Governance	45
	Governance: Q&A	46
	17. What Is SOA Governance?	46
	18. How Does an Organization Get Started with SOA Governance?	49
	19. What Is the Role of Change Management?	53
	20. Does Implementation of SOA Tools and Infrastructure Equate to SOA Governance?	55
	21. Should Service Development Be Centralized in Service Centers?	57
	22. Does SOA Require Centers of Excellence, Architecture Boards, or Design Boards?	58
	23. Why Do Organizations Need to Focus on SOA Governance When There Is an Effective Enterprise Architecture Activity?	61
	24. Is SOA Governance Required for SOA Projects to Be Successful?	63
	25. How Can You Measure Whether SOA Governance Is Effective?	64
	26. What Is the Difference Between Design-Time and Runtime Governance?	65
	27. What Are Common Pitfalls of SOA Governance?	66
	Governance: Key Concepts	67

Chapter 4	Information	69
	Information: Q&A	70
	28. What Is the Relationship Between Information Architecture and SOA?	70
	29. What Are Information Services?	71
	30. How Are Information Services Classified?	72
	31. Do Information Services Differ from Other Services?	74
	32. How Should Information Services Be Identified?	76
	33. When Should Information Services Perform Create, Read, Update, and Delete (CRUD) Operations?	77
	34. Are Enterprise Information Models Required for Effective SOA Implementations?	78
	35. What Is a Canonical Message Model?	80
	36. How Should a Canonical Message Model Be Created?	82
	37. Can SOA Improve Data Quality?	83
	38. What Are the Common Pitfalls with Information Architecture and SOA?	84
	Information: Key Concepts	85
	Index	87

Content Excerpted from
Pearson Technology Group by IBM.
Not for Resale or Distribution.

Introduction

You will never stub your toe standing still. The faster you go, the more chance there is of stubbing your toe, but the more chance you have of getting somewhere.

—Charles Kettering

A myth abounds that ostriches hide their head in the sand when frightened, and that same behavior is often attributed to anyone who foolishly ignores problems while hoping those problems magically vanish. The ostrich does many things, but hiding its head in the sand is not one of them. IT departments do many things, and hiding their heads in the sand is unfortunately one of them.

IT departments face many challenges, one of the biggest being that they spend a substantial part of their resources on running the business rather than changing the business. That is, they spend substantially more money on maintenance than on innovation. And this particular problem is getting worse and cannot be improved upon by inertia or standing still. Instead, change is required, and this book covers how to adopt *service-oriented architecture* (SOA) as a change agent (and deal with the inevitable stubbed toes along the way).

Several forces and events contribute to inefficiencies and higher costs for many IT departments: acquisitions, fiefdoms, technology zealots, infrastructures built over time without a roadmap, financial measurements that incent IT to be cheaper rather than more effective, poor application portfolio management, and ineffective architectural policies. The effects of such inhibit IT departments' ability to accelerate or improve time to market for new business capabilities. SOA can make a significant and positive difference, but you must

understand that this is a process, a true journey. After all, technology implementation by itself does not guarantee business agility.

About This Book

According to your needs and familiarity with SOA, you can use this book as a textbook, quick reference guide, or a tutorial. You do not need to read the book sequentially. In fact, you can start at any chapter and even jump between chapters to learn about the areas that interest you, and you can do so without losing context/continuity.

This book inventories challenging questions from business and IT stakeholders and provides corresponding answers. Where appropriate, the answers are prescriptive. Although, in this book, we attempt to exhaustively anticipate your questions and provide readily understandable answers, we also provide an outside forum for you to ask, in your own words, any questions we might have failed to address. You can access this forum at www.100Questions.info. We invite you to continue our SOA dialogue there.

Questions are numbered sequentially from 1 to 100 throughout the entire book.

Intended Audience

This book is intended for executives, managers, IT architects, business architects, business analysts, line-of-business (LOB) managers, and students who want to understand the basic and complex concepts of SOA and the business and technology rationales for developing and implementing SOA.

For example, readers might include the following:

- LOB/product managers who wonder what SOA has to do with the business
- Business executives/stakeholders who want to know how to make new development projects have built-in flexibility and sustained agility
- Business/IT stakeholders who want to know what they need to do differently to make systems more agile

- Architects tasked with a transformation initiative or project and who want to understand how or whether SOA can be applied
- Architects who want to understand how to build a system for change so that the application is not difficult to change three or five years after its initial production deployment
- Enterprise architects who want to be more effective at creating adaptive and usable enterprise architectures
- Students and others who want to know the facts about SOA

How This Book Is Organized

This book is organized in such a way that you can browse and easily find topics of interest. The chapters themselves address specific domains of concern about SOA in the business/IT world, as follows:

Chapter 1, “SOA Basics”—This chapter defines *SOA* and *service orientation*. It also examines several myths and misconceptions that prevail in the marketplace about SOA.

Chapter 2, “Organization”—This chapter discusses the technology and organizational roadblocks that impede forward motion in SOA adoption. The chapter also examines the relationships between business and IT and how they collaborate for SOA.

Chapter 3, “Governance”—This chapter addresses the hot topic of governance, including why it is important and its impact on achieving business results with SOA adoption. The chapter answers questions about governance, adoption steps, how to get started, and how to communicate the SOA journey.

Chapter 4, “Information”—This chapter covers how information, data architecture, and management support SOA. Concepts addressed in this chapter include information as a service, canonical models, and message models.

At the end of each chapter, we address common pitfalls and how to avoid them. After all, before organizations can take preemptive measures to avoid missteps in SOA adoptions and initiatives, they must understand where others are making mistakes.

Content Excerpted from
Pearson Technology Group by IBM.
Not for Resale or Distribution.

1

SOA Basics

Delusions, errors, and lies are like huge, gaudy vessels, the rafters of which are rotten and worm-eaten, and those who embark in them are fated to be shipwrecked.

—Buddha

Service-oriented architecture (SOA) is defined in a number of ways, but not all definitions are equal, and not all definitions are complete. Instead of just providing another definition of SOA, this chapter describes the basic building blocks of SOA and looks at the value proposition of SOA from key stakeholder perspectives. Besides covering the basic building blocks of SOA, its DNA, and the value propositions of adopting SOA and its ultimate utility, this chapter describes what makes an implementation an SOA deployment. Specifically, this chapter addresses the following questions:

1. What is SOA?
2. Is SOA an architectural style?
3. What are fundamental constructs (the DNA) of SOA?
4. What is the difference between a Web Service and an SOA service?
5. What makes a project an SOA implementation?

SOA Basics: Q&A

1. What Is SOA?

Numerous vendors, application providers, system integrators, architects, authors, analysts firms, and standards bodies provide

definitions of SOA. The definitions of SOA are diverse. Most are complementary and do not conflict with each other. SOA has a variety of definitions because the definition is often tuned to a specific audience, as explaining SOA to a CEO is different from explaining SOA to a programmer. The term *service orientation* is often used synonymously with SOA, but just like SOA it has a wide range of interpretations. Service orientation is broader and represents a way of thinking about services in the context of business and IT. This book makes no distinction between SOA and service orientation and in some cases may use the two terms synonymously.

An agreed-upon definition for SOA eludes the industry. Anyone reading Wikipedia's definition page for SOA will see the challenges of trying to gain consensus on an SOA definition. Standards bodies, the OASIS group, and the Open Group have provided complementary but different SOA definitions. Presented with a blank sheet of paper, an artist sees a canvas. A poet might fill it with verse. An engineer seizes the opportunity to make a paper plane. Kids may see it as a future pile of spit wads. SOA is that blank sheet of paper.

To the *chief information officer* (CIO), SOA is a journey that promises to reduce the lifetime cost of the application portfolio, maximize *return on investment* (ROI) in both application and technology resources, and reduce lead times in delivering solutions to the business.

To the business executive, SOA is a set of services that can be exposed to their customers, partners, and other parts of the organization. Business capabilities, function, and business logic can be combined and recombined to serve the needs of the business now and tomorrow. Applications serve the business because they are composed of services that can be quickly modified or redeployed in new business contexts, allowing the business to quickly respond to changing customer needs, business opportunities, and market conditions.

To the business analyst, SOA is a way of unlocking value, because business processes are no longer locked in application silos. Applications no longer operate as inhibitors to changing business needs.

To the chief architect or enterprise architect, SOA is a means to create dynamic, highly configurable and collaborative applications built for change. SOA reduces IT complexity and rigidity. SOA

becomes the solution to stop the gradual entropy that makes applications brittle and difficult to change. SOA reduces lead times and costs because reduced complexity makes modifying and testing applications easier when they are structured using services.

To the IT architect, SOA is the architectural solution for integrating diverse systems by providing an architectural style that promotes loose coupling and reuse. Many IT architects think they have seen this style before with earlier architectural initiatives such as DCE, the *Distributed Computing Environment*, and CORBA, the *Common Object Request Broker Architecture*.

To the developer, SOA is a programming model or paradigm where web services and contracts becomes a dominant design for interoperability. It is a web service when it uses a *Web Service Description Language* (WSDL) or equivalent specification for describing the service. Web services enable organizations to communicate information, using messages, without intimate knowledge of each other's IT systems.

Delivering on the promises of SOA (improved business agility, maximized ROI, reduced IT complexity and rigidity, reduced costs, reduced lead times, reduced risk, new opportunities to deliver value, increased participation in value networks, and incremental implementation) requires you take a holistic view of SOA. If we limit the view of SOA to a single stakeholder (e.g., IT architect, developer, or business analyst) the benefits will not accrue because SOA then just becomes one in a long list of overhyped technologies rather than a novel approach to building flexible business solutions.

2. Is SOA an Architectural Style?

SOA is often seen as an architectural style that has been around for years. Figure 1.1 shows the architectural style of SOA. In this scenario, a service consumer invokes or uses a service. The service consumer uses the service description to obtain necessary information about the provider service (e.g., account service) to be consumed. The service description provides the binding information so the consumer can connect to the service, and the description identifies the various operations (e.g., open or close account) available from the

provider service. A broker can be used to find the service using a registry that houses information about the service and its location.

In Figure 1.1, it is difficult to determine how the architecture style of SOA enables the strategic benefits of SOA, such as lowering the lifetime cost of an application or bringing faster time to market or making applications resilient to change. SOA as an architectural style often makes an SOA project solely an IT endeavor where the strategic business benefits of SOA no longer become the focus or measured outcomes. Benefits of process flexibility, time-to-market savings, lower costs, and others can be achieved with SOA, but only if we holistically adopt all stakeholder views of SOA and its application and pursue SOA adoption accordingly. When pundits, architects, consultants, or executives define SOA as a pure technology play or as solely an architectural style, they relegate it to the realm of IT science projects, overhyped technologies, and a marketing strategy *rather than a novel approach to building flexible business solutions*.

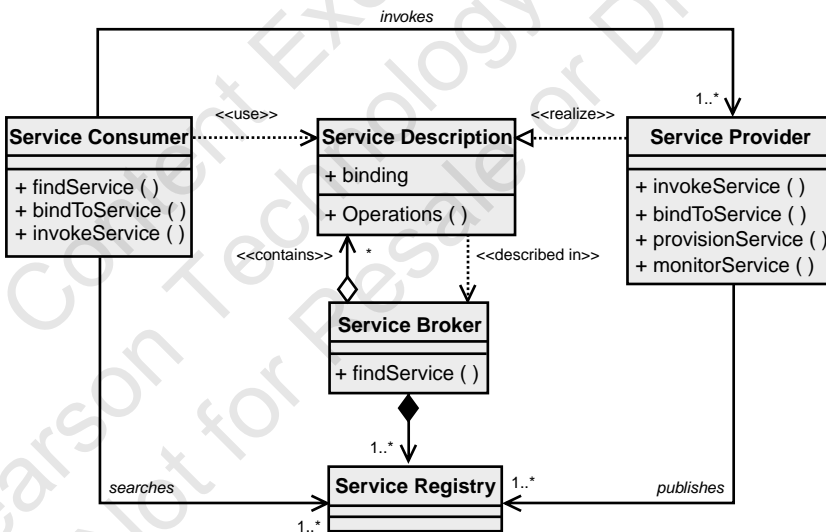


Figure 1.1 SOA as an architecture style

An understanding of SOA is enhanced with the next question and answer. By looking at the SOA building blocks of SOA, you can gain a fuller understanding of what SOA is and how to realize its promised benefits.

3. What Are the Fundamental Constructs (the DNA) of SOA?

The most basic construct or building block of SOA is a service. Software engineering over the years has evolved from procedural to structured programming to object-oriented programming to component-based development and now to service oriented. Figure 1.2 illustrates the different levels of abstraction from objects to services. Each evolution of abstraction builds on the previous, and SOA embraces the best practices of object and component development.

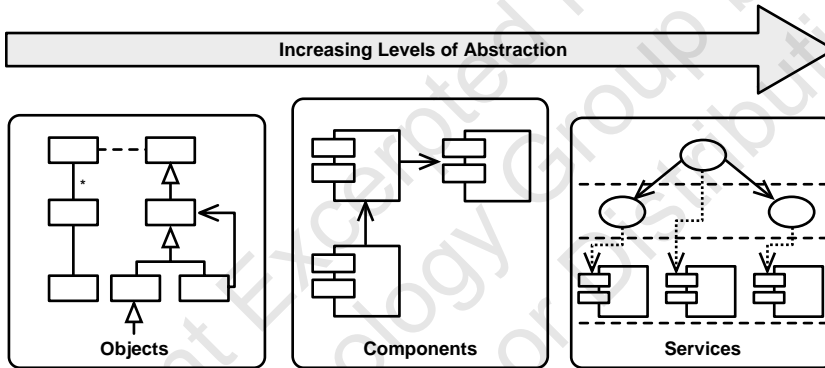


Figure 1.2 Levels of abstraction

To see architectural style of SOA, refer to Figure 1.1. That illustration shows the fundamental constructs of SOA, such as the service consumer and the service provider and their relationship. The consumer invokes a service, the business functionality, by contract. The provider of the service defines the contract as a service description. An intermediary, such as a broker, uses a registry to find or search for published services. Service consumer, service provider, service description, service broker, and a registry are all part of the DNA of SOA.

A service in SOA is the logical, self-contained business function. Services in SOA have the following attributes:

- **Stateless:** SOA services neither remember the last thing they were asked to do nor do they care what the next is. Services are

not dependent on the context or state of other services, only on their functionality. Talking on the telephone is stateful, whereas posting a letter is stateless. The World Wide Web provides an excellent example, where each request from a user for a web page or URL results in the requested pages being served, but without the web server remembering the request later. Each request or communication is discrete and unrelated to requests that precede or follow it.

- **Discoverable:** A service must be discoverable by potential consumers of the service. After all, if a service is not known to exist, it is unlikely ever to be used. Services are published or exposed by service providers in the SOA service directory, from which they are discovered and invoked by service consumers.
- **Self-describing:** The SOA service interface describes, exposes, and provides an entry point to the service. The interface contains all the information a service consumer needs to discover and connect to the service, without ever requiring the consumer to understand (or even see) the technical implementation details.
- **Composable:** SOA services are, by nature, composite. They can be composed from other services and, in turn, can be combined with other services to compose new business solutions.
- **Loose coupling:** Loose coupling allows the concerns of application features to be separated into independent pieces. This separation of concern provides a mechanism for one service to call another without being tightly bound to it. Separation of concerns is achieved by establishing boundaries, where a boundary is any logical or physical separation that delineates a given set of responsibilities. For example, an account service has open account, authorization, and audit features representing delineations of responsibilities and three separations of concerns.
- **Governed by policy:** Services are built by contract. Relationships between services (and between services and service domains) are governed by policies and *service-level agreements* (SLAs), promoting process consistency and reducing complexity.
- **Independent location, language, and protocol:** Services are designed to be location transparent and protocol/platform

independent (generally speaking, accessible by any authorized user, on any platform, from any location).

In addition, services in a service-oriented architecture typically have the following characteristics:

- **Coarse-grained:** Services are typically coarse-grained business functions. Granularity is a statement of functional richness for a service—the more coarse-grained a service is, the richer the function offered by the service. Coarse-grained services reduce complexity for system developers by limiting the steps necessary to fulfill a given business function, and they reduce strain on system resources by limiting the “chattiness” of the electronic conversation. Applications by nature are coarse-grained because they encompass a large set of functionality; the components that comprise applications would be fine-grained. Similarly, within an application, a service such as “get account information” (which returns name, account number, and address) could be described as coarse-grained, whereas a service to “get account number” could be described as fine-grained.
- **Asynchronous:** Asynchronous communication is not required of an SOA service, but it does increase system scalability through asynchronous behavior and messaging techniques. Unpredictable network latency and high communications costs can slow response times in an SOA environment, due to the distributed nature of services. Asynchronous behavior and messaging allow a service to issue a service request and then continue processing until the service provider returns a response.

Viewed from the top down, SOA comprises the following constructs, as illustrated in Figure 1.3: consumer, business processes, services, components, information, rules, and policies. Consumers allow invocation or composition of services at the consumer layer through social software, mashups, business processes, or other systems. Business processes represent the flows of activities required to complete a business process; they are compositions of services targeted to achieve business goals. Services are the main structuring element required by a service consumer and are provided by the service provider. Services offer functionality and quality of service,

both of which are externalized within service descriptions and policies. Services can be composed of other services, thus making them composite services. Components realize not only the functionality of the services they expose but also ensure their quality of service. Information flows between the layers (for example, consumer, process, and service) and within a layer. Lastly, rules and policies exist for services, components, and flows.

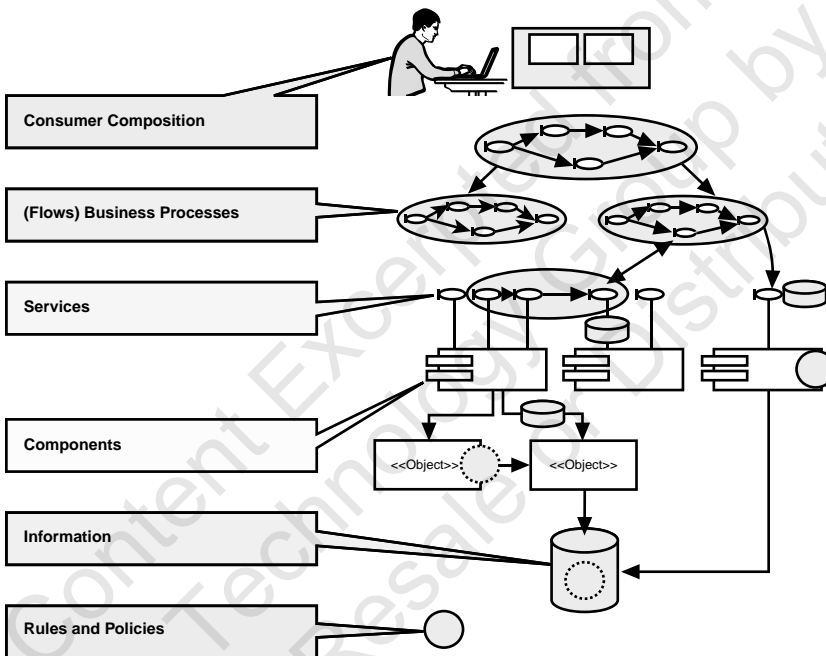


Figure 1.3 Top-down view of SOA constructs

Although objects are illustrated in Figure 1.3, the word *object* does not imply an implementation of object orientation, because the object can easily be a procedural subroutine implemented in a multitude of languages as easily as it can be implemented in an object-oriented programming language. SOA must have services and components that realize the services. Processes or flows may string services together to fulfill a step or activity of a business process. For example a transfer of funds service may string together both a debit and credit account service.

There is also a technology view of SOA. Technology enables SOA, makes it efficient, and optimizes the implementation, but SOA is not defined by the technologies chosen for implementation. Instead, SOA is defined by providing a uniform means to offer, discover, interact with, and use capabilities (services) to produce desired effects consistent with measurable expectations.

The major technologies associated with SOA include business-focused tools, software construction tools, and middleware technologies. Figure 1.4 illustrates the basic technology building blocks for SOA. Tools are required for SOA addressing design-time and run-time scenarios. Business stakeholders use business-focused tools for modeling and analysis of business processes and flows, and they will also use business activity monitoring technology to gain insights into business performance of processes and workflow. IT practitioners use a set of tools for development of business applications and for managing the operating environment addressing integration, monitoring, and security.

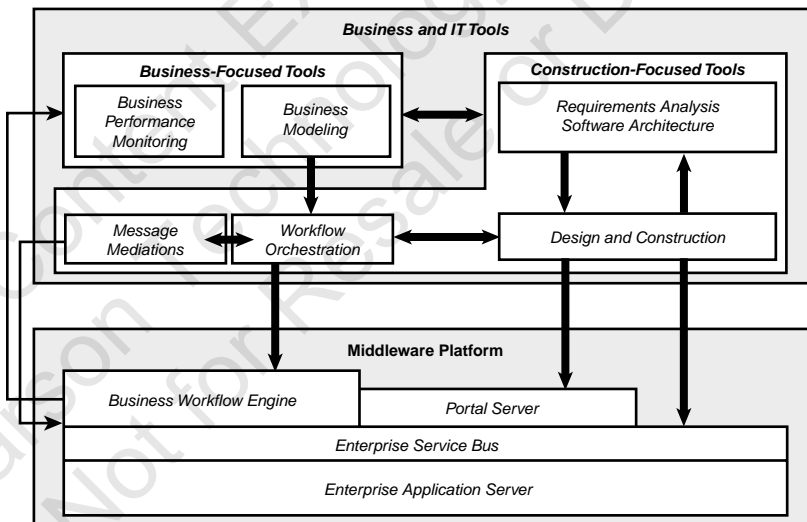


Figure 1.4 Business and IT tools for SOA

The DNA of SOA will most likely be further investigated and defined by standards groups actively involved in defining an SOA

ontology. For example, see www.opengroup.org/projects/soa-ontology/. Such an ontology will address SOA key concepts, including services, service contracts, service interfaces, composition (orchestration, choreography, and collaboration), processes, service compositions, policies, and events. Each of these makes up the DNA of SOA.

4. What Is the Difference Between a Web Service and an SOA Service?

The distinction between business services or SOA services versus a web service is not often articulated, and many equate the two as being the same. SOA services can be realized as web services, but not all web services are equal to SOA services. Web services represent the use of both a published standard and a set of technologies for invocation and interoperability. SOA services are services that fulfill a key step or activity of a business process and can be described as business services and are often exposed as web services.

Figure 1.3 illustrates both an SOA service and a web service. The picture shows the difference between SOA and web services at runtime (i.e., implementation level) and at design time. The web service is illustrated on the right side of Figure 1.5, specifically the *Web Services Description Language* (WSDL) and its attributes such as port types and operations. The attribute that makes it a web service is the use of WSDL or equivalent.

In design, we identify and specify a service that provides the design, or we identify and specify interfaces that include method specifications. The combination of the definition of the method and the interface at design time is what we refer to as a *service* from an SOA perspective. Use cases can be used to capture the functional requirements for a service. Figure 1.5 contrasts the differences between a service in SOA and a web service. Both SOA services and web services are part of the DNA of SOA.

In an SOA, business processes, activities, and workflow are broken down into constituent functional elements called services. They can be accessed and used directly by applications, or they can be mixed and matched with other services to create new business capabilities. Business services or SOA services are reusable business

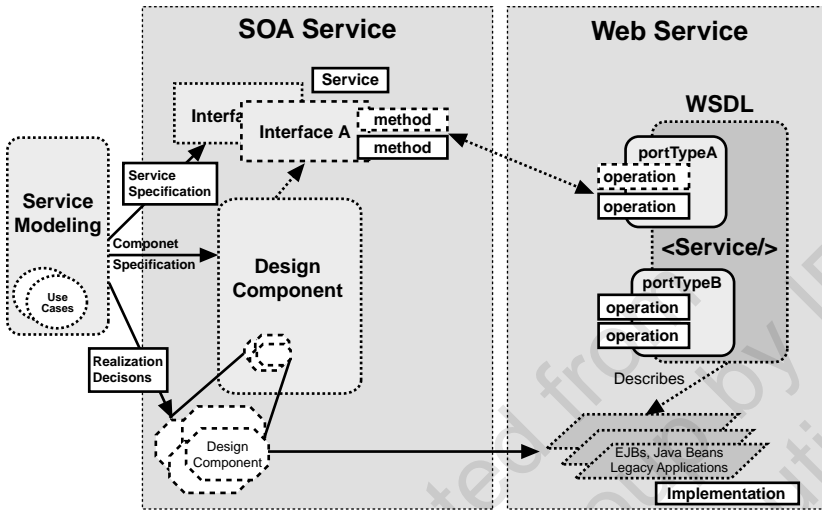


Figure 1.5 SOA service and web service

capabilities. Examples in banking include open account or change address. For transportation, it might be get reservation or hold reservation, and with loan processing, get loan, apply for loan, and update address are examples of business services. Business processes are also key constructs of SOA, part of its DNA.

5. What Makes a Project an SOA Implementation?

The deployment of services makes a project an SOA implementation, where a *service* is defined in the preceding answer as a web service or an SOA service. The use of the *Web Service Description Language* (WSDL) or equivalent makes a service a web service. An SOA service must satisfy the criteria described in the Answer 2; namely, an SOA service must be stateless; discoverable; self-describing; composable; loosely coupled; governed; and independent of location, language, or protocol. That is, the use of services alone makes the project or implementation an SOA implementation. However, an SOA implementation may not accrue the desired benefits of SOA around cost savings, reuse, time to market, or flexibility.

Services can have different levels of maturity. For example, services can be ad hoc in their design and implementation where a

WSDL façade is implemented to make function accessible to other systems or applications. Services can also be architected where service modeling and governance are used to maximize service reuse.

The implementation of SOA technologies without a deployment of one or more services could also be defined as an SOA implementation. This would be atypical because middleware and infrastructure implementations (e.g., a registry or enterprise service bus) would be implemented in conjunction with the deployment of services.

Just as services have different levels of maturity, so do SOA adoptions within an organization. Some SOA adoptions require a program of projects to address a journey of increasing maturity to achieve strategic SOA goals of building systems for change, infusing flexibility as an attribute of systems, or reducing the lifetime costs of applications and infrastructure. In this case, the program comprises a series of SOA projects that incrementally raise the maturity of SOA in an organization and along the way enable the realization of the strategic SOA benefits.

Often, because of overselling of SOA, organization leaders, managers, and executives wrongly believe that the benefits of SOA automatically accrue when an SOA implementation occurs. SOA has varied and diverse definitions, and hence its implementations are equally varied. So, organizations seeking to accrue any of the promised benefits of SOA must do more than focus on SOA implementations. That is, each expected benefit of SOA requires a different level of SOA maturity. For example, if the goal is only to reduce the cycle time of a business process that deals with external partners, exposing web services may be the only necessary SOA adoption. However, if the business goal is to reduce time to market for new products, this requires a broader adoption of SOA that addresses reusable services, structuring of applications using services, improving integration using services, and aspects of SOA governance to address service sharing, funding, and ownership.

SOA Basics: Key Concepts

This chapter's answers emphasized the utility of SOA and how to accrue its strategic and tactical benefits, instead of just providing an agreed-upon definition. However, looking at the definition through the lens of the different stakeholders provides a comprehensive view

of what SOA is highlights the various potentials of SOA. The DNA of SOA comprises service consumers, business processes, services, service descriptions, components, information, rules, policies, web services, technologies (e.g., registries and brokers), and tools that address business and IT domains.

As you learned in this chapter, SOA implementations are as varied as SOA definitions, and the benefits that accrue depend on the maturity of SOA adoption within an organization. Organizations and executives who expect to accrue strategic benefits of SOA will need to treat SOA adoption as a journey realized incrementally by project (not as tactical goals, where a project might be sufficient). The next chapter answers questions that business leaders and executives ask about SOA.

Content Excerpted from IBM.
Pearson Technology Group
Not for Resale or Distribution.

Content Excerpted from
Pearson Technology Group by IBM.
Not for Resale or Distribution.

2

Organization

There is a potentially infectious condition inside virtually all organizations that can cause more damage than economic downturns, management upheavals, or global business shifts. Until now it has no name. But this condition has been an enormous problem in all facets of business...

I call it the “fiefdom syndrome,” and it happens to all organizations, large and small, profit and non profit. It occurs at the individual level as well. And it can significantly decrease an individual’s, and a company’s, effectiveness.

The fiefdom syndrome stems from the inclination of managers and employees to become fixated on their own activities, their own careers, their own territory or turf to the detriment of those around them.

People who tend to hoard resources. They are determined to do things their own way, often duplicating or complicating what should be streamlined throughout the company, leading to runaway costs, increased bureaucracy, and slower response times.

—Robert J Herbold, *The Fiefdom Syndrome*

By now, everyone who has launched SOA projects or attended SOA conferences or read the numerous articles and book knows that organizational issues are as important as technical issues for achieving many SOA goals. Many of the organizational issues that impede meeting SOA goals relate to governance, its presence or absence; but another set of issues relates to behavior and culture—organizational issues. Several of these issues reflect the fiefdom syndrome, where

lines of business, divisions, or departments within a company avoid sharing, battle against standardization, and resist change. The fiefdom syndrome can be seen as politics or turf wars, but for many trying to make SOA promises real, addressing the fiefdom syndrome seems like trying to boil an ocean. Anticipating organizational issues and preparing ahead of time with effective approaches is essential to success in SOA. Organizational issues matter, and this chapter provides answers for the various questions we have addressed on several projects:

6. How does business/IT alignment change because of SOA?
7. Which joint business/IT processes change because of SOA?
8. What organizational structures should be established for SOA?
9. What is the role of organizational change management to SOA?
10. How can organizational barriers to SOA success be removed?
11. How should organizations address funding for services?
12. How should organizations address prioritization for shared services?
13. What is the value of classifying services?
14. Who owns service reuse?
15. What are service owners?
16. What are the common organizational pitfalls when adopting SOA?

Organization: Q&A

6. How Does Business/IT Alignment Change Because of SOA?

The alignment between business and IT in most organizations lies on a continuum between highly collaborative and tense. This is not unusual given that all relationships involving people, money, and commitments can create a highly collaborative environment or one of mistrust and tension. SOA provides the opportunity for a middle ground, where both the business and IT get what they have sought:

business gets new levels of flexibility, while IT reaps the rewards of a uniform architecture built for change. Application development by the consumers of services can occur without waiting in long IT development queues by the provider of services; the decoupling of the IT side from the business side through services and a federated architecture makes this possible. SOA changes and improves the relationship between business and IT as a result of shared services. For bank debit or credit accounts, transfer funds are built once as reusable services regardless of whether access is through a call center, web interface, or mobile device. For insurance companies, submit loan application or perform claims adjudication; for telecommunication carriers, get location information; and across all industries, get account information and update customer address are not functions built many times because of silo applications, silo organizations, or different access channels, but shared services that are built once and reused. The strategy shifts to a reuse, buy and build versus buy and build.

Improving the relationship between business and IT does not occur magically or without planning. Figure 2.1 illustrates the general range of business and IT relationships, where unified is a desired state.

In the unified state, people come together as teams for collaboration, problem solving, portfolio management, project prioritization, and governance. Stakeholders don't wear hats called business or IT when they meet because they know these labels don't reflect the breadth of their contributions. Business stakeholders increasingly provide suggestions about IT, and IT stakeholders provide business insights; each side plays an integral partnership role in the success of the organization.

In the synced state, teams are formed to work on problems. However, the team often dissolves when the problem is considered "solved." Organizational reporting hierarchies often take precedence over domain knowledge in selecting team members. Governance is maturing in the sync state and there is a lot of conversation on how to improve and align business and IT. Organizational change is discussed but material changes have yet to take effect that change behavior between business and IT.

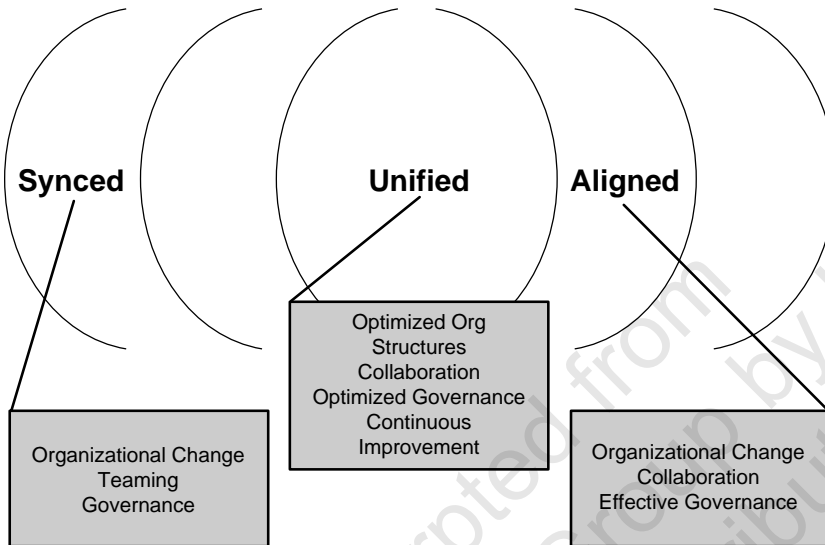


Figure 2.1 Business and IT relationship states

A collaborative environment with effective governance characterizes the aligned state. Business and IT alignment is a mantra often echoed when looking at how to improve business and IT relationships. Using a standards- and service-based approach where a service repository can be used as a central authority (much like databases do for information) changes business and IT relationships into a service. Like data that becomes a common language, a business language develops between business and IT. Regardless of whether organizations find themselves at the unified, aligned, or sync level of maturity, when services becomes a business and IT term, there is an opportunity for improved dialogue between business and IT. In addition, collaboration and sharing increase when designing and constructing the shared services.

Unified is an optimal state for business and IT relationships, it represents a convergence of business and IT, a partnership. The relationships of a unified state are not born without hard labor, but the result is a highly collaborative environment built on trust. Organizational structures are continually optimized to create operational dexterity that allows organizations to be more agile. Governance is active, effective, and tweaked continuously using measurements and feedback loops. Continuous improvement to measure, tweak, and

monitor is a way of life. Reaching this unified state of relationship between business and IT is a goal of SOA adoption that occurs incrementally from sync to align to unify.

7. Which Joint Business/IT Processes Change Because of SOA?

Several joint business/IT processes change as a result of SOA including: governance, portfolio management, strategic planning, managing investments, requirements gathering, and project prioritization. These changes are as follows:

- Governance is collaborative. Both business and IT stakeholders are inserted into existing processes related to how projects are funded, and when projects are built, bought, extended, or reused as business functionality. Each and every governance process should be examined with an eye toward establishing/enhancing collaborative roles between business and IT.
- Portfolio management is in place to avoid the proliferation of services and applications. “Less is more” is the mantra, but this requires active management where both the business stakeholders responsible for the business operations and the IT stakeholders responsible for IT automation sit together to discuss how to extend, retire, provision, or reuse an existing services portfolio of shared business services. Reuse becomes a priority versus buying or building from scratch.
- Strategic planning addresses sharing across the organization for increased business operational flexibility. Sharing of services and its enabling infrastructure occurs regardless of whether the organization represents a centralized or decentralized IT delivery model. Issues pertaining to standardized business processes are addressed as the return of investment of the overall organization is favored over the investment of a single business unit.
- Managing funding for how sharing of services is either promoted or governed. In some cases, “business as usual” cost allocation and funding models work, and for other organizations these models must be enhanced. Success of shared services should justify increased investments.

- Requirements gathering also change how business and IT interact and relate when specifying requirements. Practices that promote application silos give way to practices that promote the strategy of “build once and reuse.” This requires a twofold approach: providing a view into what can be reused at the business level during requirements gathering and moving away from specifying requirements as functional domains, which is largely done today with use case modeling.
- Project prioritization takes into consideration the shared functionality necessary for the on-time, on-budget delivery of projects. Globalization, increased competition, and empowered customers force a choice about what to do first. Services such as applications must be part of any prioritization scheme.

Each of these changes occurs incrementally based on the strategic and tactical goals for adopting SOA. Meeting the strategic goals of SOA ultimately, and for the promises of SOA to be fulfilled, requires reaching the unified stage depicted in Figure 2.1.

8. *What Organization Structures Should Be Established for SOA?*

Most organizations are structured to support product lines or vertical business units with IT organized accordingly, often with application teams aligned to the vertical lines of business. People, skills, and budgets are focused on discrete projects prioritized by the line of business. At the same time, an increasing number of projects need to share a business function created in another line of business. For example, in banking there may be divisions of retail, wholesale, credit cards, and loans. Shared information about a customer and shared business capability such as update address or get account information are needed by retail, wholesale, credit card, and loans. Different lines of business, different departments need access to discrete units of business functionality without building bridges (i.e., interfaces) or sitting in development queues waiting for access to be delivered. That is, shared services are needed and on the rise, and how companies organize themselves can make a difference.

Figure 2.2 illustrates a typical organizational structure in most large companies that have both business units and a separate IT department. Business units or lines of business typically have an application development team assigned to support their business unit residing in the IT department. In some organizations, a hybrid may exist where development resources are found both in the business unit and IT. Sitting in the IT department are often relationship managers, executives, who report to the CIO and who have a dotted-line report to a business unit executive who is peer to the CIO. This relationship manager has a deep understanding of the business unit and is responsible for the IT automation needs of their business units. Business analysts or business architects often report into this area. Application development teams are in the IT department. These are often silo and centralized teams that provide support, database administration, network engineering, infrastructure, operations, security, and in some organizations, enterprise architecture (EA).

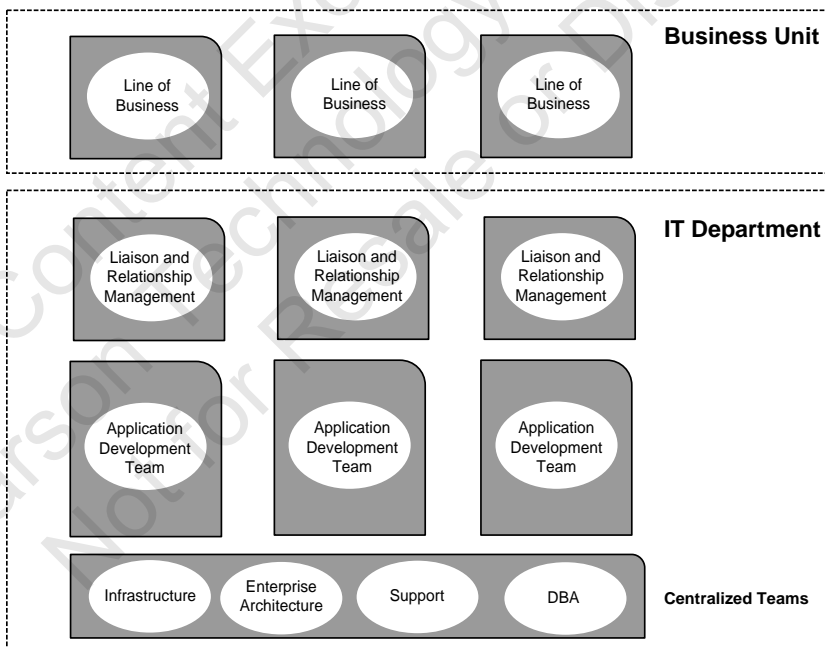


Figure 2.2 Typical IT organizational structure

The debate for many organizations about what should change for SOA often centers on whether there should be a centralized team for service development and what the role of an EA team is. Centralized teams for service development may make sense on a transient or permanent basis. Often, organizations have limited skills and resources for the architecture, design, and building of services, and centralizing this capability under a shared pool works to resolve this resource constraint. However, conflict can arise between the consumer and provider roles for services. That is, the domain knowledge to build the service is often found within the current application development teams supporting the business units, yet the development role has been removed from this team and centralized. Accountability issues ensue as the consumer becomes the application team and the provider becomes the centralized team. Finger pointing can occur around requirements and delivery as application teams have a major responsibility removed, which is building out the functionality required to support their business units.

A distinction exists between organizations pursuing factory models versus organizations centralizing the development of services. Factory models are often desired because organizations want to take advantage of a flat world and the advantages of a 24x7 clock or a cheaper work force. Examples are seen as companies move work to labor pools that are available with lower labor costs. In service development, factory models are used to enable faster and less-expensive service development. In both a factory model and a centralized service delivery model, clearly defined roles and responsibilities regarding the service provider and consumer are needed. Typically, factory models work best when only limited aspects of the service life cycle are assigned to the factory: programming and unit testing.

Organizations implementing a shared service development model want to avoid creating a fragmented project accountability model with unclear provider and consumer roles. In most cases, because of the domain knowledge and direct access to the service consumers, it makes sense for the provider role to be accountable for all aspects of service delivery. Application teams should have a provider role, and

any fragmentation or diffusion of this role using centralized teams should be temporary or account for how project delivery accountability will be resolved so as not to create an inefficiency in the development of shared services. Understanding and defining the role of service owners can help to resolve these issues because the service owner has a provider role with distinct responsibilities.

Shared service development models often fail because of fragmented and shared accountability on the responsibilities for creating a reusable service. Often the “blame game” becomes prominent, as the shared service development team does not have project accountability. The shared service team may blame the consumer (aka application) for not providing adequate requirements, domain knowledge, or any number of reasons. It’s one of the main reasons why applications dislike using shared services; they give up control. It is crucial that there be a project accountability that reaches across all groups, and in which slippages are known by all immediately, with corresponding risk mitigation. If shared service development models are used, both groups must “have skin in the game,” and operate under a single project management structure.

A modified IT organization structure for shared services is depicted in Figure 2.3, which shows two major changes to the traditional IT model: a business-focused role for an EA group and a centralized integration center. The EA team in many organizations today is largely IT focused and invisible to the business units. In the SOA world, the EA team must evolve from a primarily technical focus to a fused business and technical focus. The EA team is responsible for enterprise architecture, working closely with the business and IT as a part of that EA responsibility to identify and promote shared services across the enterprise that can be used across multiple business units. EA also works closely with the liaison and relationship executive to understand business processes that span multiple units and promotes the sharing of business process design and their corresponding shared services, rules, and information. Standards for services are also a key responsibility of the EA team.

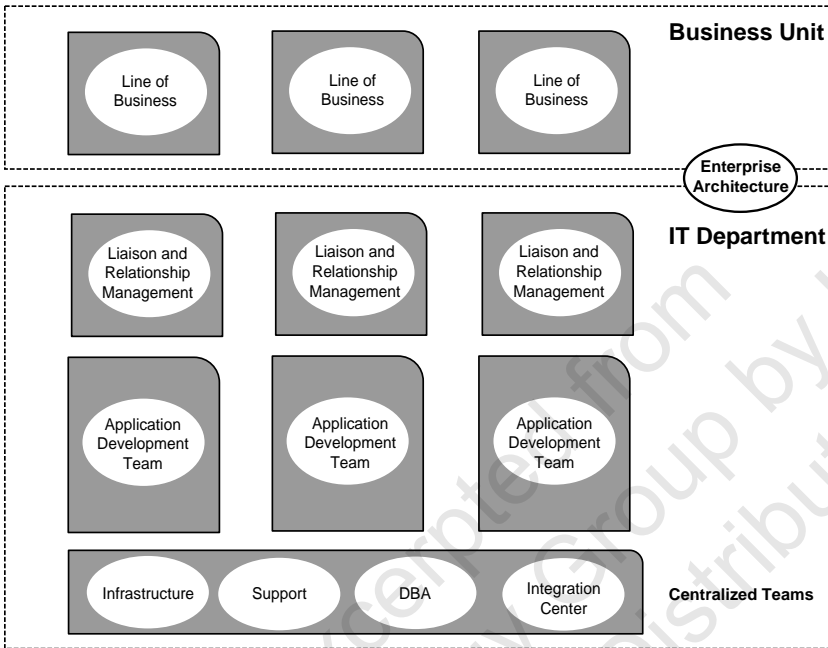


Figure 2.3 Modified IT organizational structure for shared services

The centralized integration center is responsible for addressing the business unit and their corresponding application team's vertical and horizontal integration needs using an *enterprise service bus* (ESB). The integration center has responsibility for architecture, design, development, and implementation of an ESB. This includes the development of message flows, mediation, routing, and transformation using the ESB and a registry. The integration center collaborates with the application development teams for service development and reuse. This collaboration is shown in Figure 2.4. The integration center defines and implements standards, guidelines, and processes for the ESB and registry. Facilitation of all aspects of ESB governance rests with the integration center working closely with application development teams. The integration center and application teams have joint responsibility for understanding and developing requirements for services. The application teams, possibly including a shared service application team as discussed previously, develop business services for use by the various business units and work with the integration team to satisfy integration needs within

their silos (vertical) or integration needs that span business units (horizontal). Application teams also develop the business applications that compose or orchestrate or otherwise use the services. The integration center develops the necessary mediation, routing, or transformation features or services and publishes all services using the registry so that they are easily and readily accessible using an ESB.

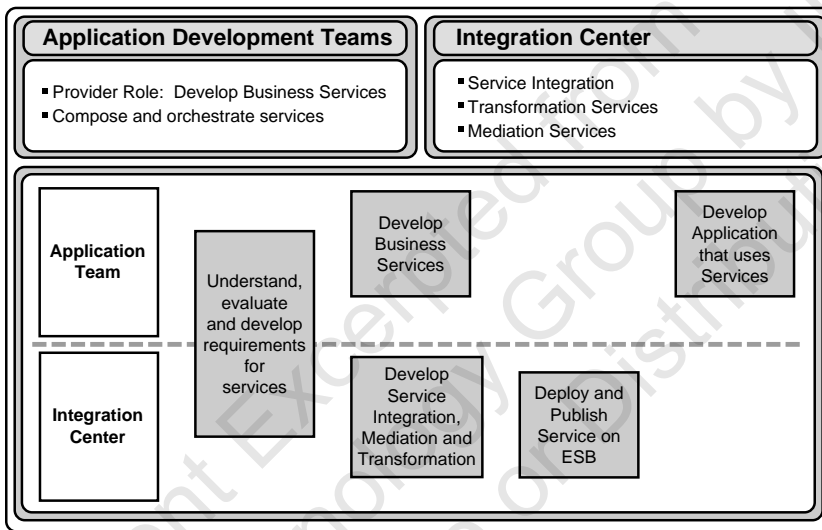


Figure 2.4 Business and IT relationship

Three organization models available for the integration center need to be evaluated against change management goals for the organization, people, and process. One option is to pool resources without concern for vertical domain knowledge or focus. The disadvantages of this option are the team may not be able to be as responsive as a vertically aligned team due to lack of access to subject matter expertise. This option might also create a bottleneck of the integration center because all requests have to be threaded through this single team. Another option is to vertically align the integration center resources along business unit boundaries specializing in the subject area. However, this has the disadvantage of promoting silos, resulting in more duplication of effort and ultimately demanding more resources than other options. The third option is a hybrid, which is to

have a vertical-based structure where the subject matter expertise lives and allows for vertical areas (business units) to quickly respond to their constituent needs. In the hybrid model, the integration center has largely technical resources skilled in how to build web services, leverage the ESB, and build services for routing, transformation, and mediation. Note that these same three options and their disadvantages can be applied to a centralized service development team (discussed previously).

The integration center is akin to *database administration* (DBA) teams that have been around for decades. The message flows and supporting integration services (i.e., mediation, routing, and transformation) are similar to what a DBA function does working to create shared data models and databases. What DBAs do for data and information, integration centers do for services and their message flows. We use this analogy to illustrate that organizations already know how to make integration centers work. However, integration centers represent a change in organization models when adopting SOA, and the good news is we know they can work

9. What Is the Role of Organizational Change Management to SOA?

Organizational change management is the answer to how barriers to SOA success can be removed organizationally. Organizational change management is required when adopting new strategies, such as SOA, because it requires a change in how teams develop applications, relationships, and interactions between business units and IT and changes within IT departments as to how they interact. Figure 2.5 illustrates the elements that organizations must understand, assess, and perform to create cultural and behavioral changes necessary to fulfill many strategic SOA goals. Change management affects the organization, processes, and IT when adopting SOA. It requires that there be some strategic goals, defined at the executive level, around performance measures, efficiency, and effectiveness goals. Ultimately, an assessment must be made about what is working today in the organization and what improvements must be made, and this is based on the demands of

the strategy and marketplace and balanced by supply of resources, skills, people, and tools.

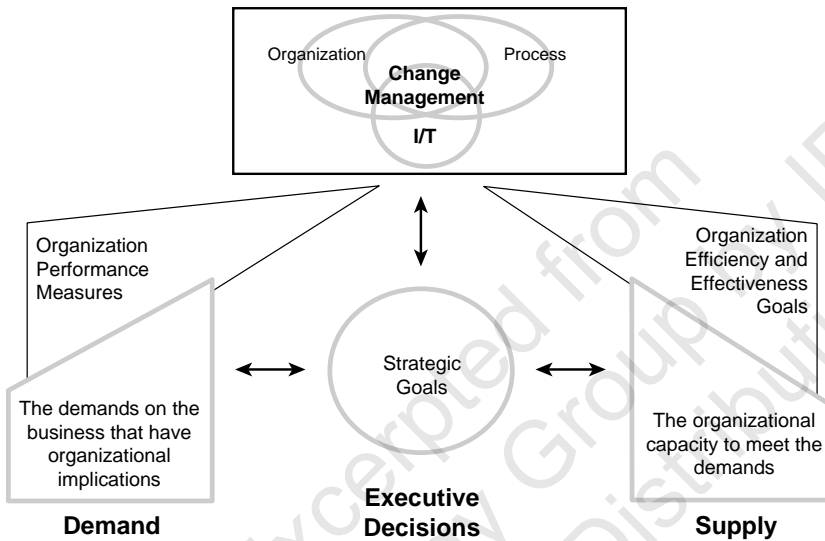


Figure 2.5 Fundamentals of organizational change management

Change management is a well-established discipline, and organizations lacking expertise in how to start, assess, and implement change management should seek outside expertise. Communications is a key aspect and success factor for any strategic objective; the stakeholders get connected and updated before, during, and after the change initiative. For example, suppose the strategic goal is to improve integration both in efficiency (e.g., reduced cost of integration and improve productivity) and effectiveness (e.g., improve flexibility and respond faster to business demands). This goal has led to the adoption of an *enterprise service bus* (ESB), registry and shared services. Many in the IT department may not understand the difference between enterprise application integration versus using an ESB for service orientation. In the business units and their liaisons, resistance may exist to sharing services because they see a potential negative impact on the reliance of other teams outside control of their vertical business unit for project delivery. Others in the organization

might not understand SOA, or they might have both negative and positive opinions about it. Still others might have seen the failure of past projects that promoted sharing of software assets. In each case, this points to the need for communication of the strategic objective before, during, and after the implementation of the ESB and its corresponding processes and organizational changes.

10. How Can Organizational Barriers to SOA Success Be Removed?

A battle rages among many circles as to who is holding SOA back, the business or IT. Of course, that depends on a lot of factors, but our experience shows that it varies by organization. In some IT departments, there is a heavy sell of SOA to the business, and it can work when there is trust and collaboration, often the result of business and IT consistently working collaboratively to produce noticeable and measurable business outcomes. In other IT departments, SOA just becomes another in a long series of lofty and unfulfilled promises. Chapter 1, “SOA Basics,” provides answers about how to sell SOA to the business. Selling SOA is not about selling SOA but about selling how to achieve specific strategic and tactical goals of the business. Selling SOA can result in “shooting of the messenger,” and the messenger might become a pariah; so it’s advised not to sell SOA. However, thinking that expressing whether the business should be interested in SOA is the same as whether the business should be interested in Java or COBOL as a programming language is not correct. The choice of programming languages is strictly an IT concern; the adoption of SOA will require business and IT unification in using services for project scope management, requirements capture, and organizational reuse. Business stakeholders have a role in SOA adoption.

SOA is facing resistance as does all major shifts in human endeavors, and IT cannot escape this resistance. Resistance is found in leaders, executives, developers, and architects, and it’s embedded in organizational models. Change occurs one death at a time. In the case of SOA, it might be the death of common practices, organizational models, or changing roles of key people. SOA continues to mean different things to different people, and this alone creates a barrier to success. The broken promises and platitudes surrounding

SOA have been used for decades, and cynicism and caution prevail. However, this again is directly related to whether organizations find themselves with a unified, aligned, or synced relationship between business and IT.

Breaking down barriers should start with defining a vision and strategy for how SOA will make a difference for an organization. Having a shared vision is integral to breaking down organizational barriers. This strategy should be grounded with the strategic and tactical goals facing lines of business and the enterprise and be accompanied by a practical and measurable roadmap for realizing the strategy. Selling SOA to the business is not recommended. Instead, projects should be identified and sought that are suitable for realizing both tactical and strategic goals of SOA. In addition, pursuing organizational change management can also make a huge difference in removing barriers.

Our experience also shows that sharing of services can be promoted and barriers broken down by conducting a facilitated workshop with the current and potential consumers. In the workshop, ask participants whether the workshop questions are answered satisfactory and if their concerns about sharing of services have been addressed. That is, if each question of concern were resolved, would the doubters support service sharing? Our experience shows that if a workshop is held addressing service funding, service prioritization, and service ownership using the inventory of questions provided in the next three questions, then resistance to sharing services is reduced. The workshop does not presume an answer to the questions; instead, it solicits the answers from the participants, the consumers and providers of services.

11. How Should Organizations Address Funding for Services?

Many practitioners adopting SOA see the lack of funding for shared services as a major bottleneck for SOA adoption. Two issues exist when discussing funding. The first issue is funding for shared services, where the first consumer may have to pay for subsequent consumers because the service has to be designed for reuse or will be reused. The second issue is where SOA needs a kick-start for funding

related to governance, technology investments, or acquisition of new skills. Organizations have existing funding mechanisms, prior to SOA, for investment and for building business functionality; if these work and are effective, they should be modified and used for SOA adoption. So, existing and funding mechanisms that work should be used. The second issue centers on organizations that do not have investment funding for establishing governance, or shared infrastructures are disadvantaged when adopting SOA. In such cases, the shared infrastructure, governance, and other investments required for SOA adoption would need to be covered as part of project costs or buried in other investments.

Consider the issue of funding services for reuse. It is a risky business to build services with the mantra “if we build it, they will come,” which is often the underlying assumption of building services for reuse. Business functionality when developed in an application should always consider the needs of the future, and change cases that depict future needs should be captured and prioritized as part of a requirements-gathering process. However, by definition, services can be reused, as explained further in the context of applications and methods. A benefit of services is that by designing and deploying services in applications, organizations create applications that are built for change, with flexibility designed in the application architecture. This is a benefit for service regardless of whether a second consumer comes on board. However, having multiple consumers is also desired and a benefit. Having multiple consumers of a service that comprise multiple verticals is a selling point, business benefit, and design point for service development. However, in this case, service funding is not onerous because the business benefit becomes clear to both consumers. The challenge then becomes how do we convince the consumers that their needs will be met when they may have different time horizons for delivery of functional needs or different quality of service attributes (for example, underwriting needs three seconds performance but claims it is okay with five seconds)? Often the first creator of a service sees SOA as being built on the backs of their projects because they are designing services for reuse or they are designing services for use by other consumers. Both issues are best addressed as part of organizational

change management, dealing with how to make the service creator have a benefit in addition to the service consumer with the creation of services.

Dependencies on other groups exist today for application and project teams, and clearly this will increase with services, where funding is just one of many dependencies. Organizations must determine how to make this increased collaboration work, instead of abandoning the notion of moving to shared services. Service funding concerns can be addressed by answering the following questions:

- Who pays for new services?
- Who pays for a break fix?
- Who pays for service maintenance?
- Who pays for service enhancement?
- Who pays for the services' foundational aspects of hardware, software, database maintenance and any other foundational features necessary to deploy the service?

Each organization using a workshop will answer these questions differently based on their culture. For some organizations, the workshop will result in concluding that the consumer or requester of a new service should pay for the service, just as any new feature would be funded by the requesting organization. Break fix and maintenance is part of the ongoing cost of maintaining applications and services, and the provider of the service should incur this cost. Maintenance costs would not be shared because the provider operates as a software provider where some minimal staff must be kept for maintenance regardless of whether there was a second consumer. The provider is also motivated to repair any defects because the defects negatively impact the provider as well as the consumer. Service enhancement, whether it's for new business functionality or to make the service available 24x7 is also something that the requester should fund. Requestors will be motivated to use the service as-is or with enhancements because the time to market savings are present with reuse. The provider, as part of a new service or enhancement request, pays the foundational aspects of the service. This discussion and the decisions occur in the workshop.

What has not been answered is who pays for getting an organization started with understanding SOA, building out the infrastructure, acquiring new skills for service development, and applying new development and operational technologies. Some organizations have investment strategies where the funding challenge is resolved; others must sell the business to secure investment funding. Organizations have three options:

- Obtain and use investment funds.
- Fund as part of a business justification.
- Launch a skunks work project where the cost is absorbed as part of underspent funds or heroic efforts. A skunk work project is where a small team launches a project primarily for the sake of innovation. Typically, it involves volunteerism and highly motivated team members who want to prove a concept for the benefit of the organization.

Realizing reuse to achieve the efficiency and effectiveness of SOA goals described in Chapter 1 does require organizational change management and governance, and this does require investment funding. Most organizations can secure this funding with incremental SOA successes that achieve measurable and clear business benefits that are well publicized and used as goodwill for future investments in SOA adoption.

Clearly, IT departments must maintain an environment that minimizes short- and long-term costs of any initiative. Selecting the correct projects for applying SOA and provisioning the correct supporting infrastructure at the right time is also integral to effective cost management. In some cases, the technology benefits of adopting SOA are clear, and in other cases, especially when strategic goals are to be met, it's important to articulate, define, and measure the business benefits. IT departments can work with the business in the same way as prior to SOA to manage the IT portfolio and cost. Some organizations find they can reallocate maintenance funds to launch their SOA adoptions efforts.

12. How Should Organizations Address Prioritization for Shared Services?

Prioritizing services is integral to the success of SOA adoption. Consumers have multiple concerns when using a service for which they have no direct control for its development or deployment. To increase support for sharing services within an enterprise, you can ask the following questions about service prioritization:

- How do we make sure that the provider has sufficient capacity in terms of subject matter expertise to implement changes?
- How do we align priorities across multiple lines of business when one line of business depends on the service of another?
- How do we make lines of business accountable to maintain priorities/interlocks?
- Will it take longer to develop shared services?
- How do we prioritize enhancements?
- Can the provider accommodate the business request and needs based on my line- of-business schedule and budget?
- Will a request be compromised because the provider has multiple interests and consumers to satisfy?
- Is the ability to deliver faster and on time lost or impacted due to accommodating conflicting requirements, coordination challenges, or challenges in accommodating different *quality of service* (QoS) requirements, or because of dependency on an organization outside of the vertical?

The reason a service is delivered by a particular business unit should be because that unit has specific domain knowledge about the business rules, process, and information. This domain knowledge of the subject matter expert puts the vertical in the optimum position to deliver quickest. This requires that a portfolio management and project prioritization process span the enterprise versus one that is a silo. This addresses the issue of what features should be delivered first and in what sequence. It requires that service development be treated as a project, just like application development. Each of these questions has various answers, but the answers largely lie in existing project prioritization processes that can be enhanced for service development.

13. What Are Service Owners?

Service owners are the providers of services that support multiple consumers. The provider of a service should architect, design, develop, and deploy a service. Service owners are like application owners in that they facilitate the sharing of services. Assigning ownership to a service facilitates governance, and like data owners, service owners have a stewardship role for services. Deciding who the service owners are can often be accomplished by determining which business unit owns the data that the service renders. When data governance is poor, data stewardship is also lacking and the owner of the data may be undetermined. In this case, the service owner can be the primary user of the process that uses the service—the process owner. Deciding on service owners addresses the following questions:

- Who owns the service?
- What does it mean to own a service?
- What are the roles and responsibilities of a service owner?
- Does the owner have veto authority over changes to services?
- Does the service owner decide who can have access to a service?

Owning a service is being responsible for making sure the services are used and used in the way to give most benefit to the business. Owners of the service need to be directly aligned to the business owners of the associated business processes. How this is done in practice depends on the organization and its structures and goals. What is true in every case is that the focus of this role should shift to business knowledge, with support from IT, which plays a provider role for the service in its engineering and deployment. Service owners have distinct responsibilities:

- Publish and maintain software architecture for the service
- Maintain a release plan for the service
- Articulate the deployment environment to meet defined and published QoS attributes for the service
- Certify a service and publish test results and test scripts for the service
- Manage the full life cycle for the service

14. What Is the Value of Classifying Services?

Often, there is a categorization or labeling of services, whether they be described as business services, IT services, information services, or utility services. Grouping services can help with understanding the degree of reuse possibilities, the domain covered, the business area scope, or ownership model of a service. Governance and provider responsibilities can vary based on the classification. For example, business services may be assigned to business stakeholders as owners and IT services assigned to owners in the IT department. Categories can also define service domains, where a domain defines a set of related services that someone can own, maintain, support, and fund. If an organization has defined a taxonomy of services, the classification helps architects, designers, and developers understand the scope of functionality to include in a service to promote composition and reuse. Enterprise architects can help with classification, and it becomes a fast path into searching for services and leveraging architectural frameworks for the design and implementation of services.

Service classification or categorization helps to match service types to a business process model, to logical operational models, or to layered component models. For example, services could be divided into two categories of business and technical services, where business services map to business process models and technical services map to operational aspects (such as authentication and authorization engine) of the architecture.

- Business services of `createStockOrder`, `submitLoanApplication`, `renewPolicy`, `checkOrderAvailability`, `transferFunds`, `getStockPrice`
- Technical services of `validateUser`, `checkPassword`, `auditEvent`

Classifying services is useful if there is a downstream use, later in the service life cycle, for the classification and if a taxonomy for the classification is published and communicated. For example, business services is a category that could be further refined into four categories by granularity:

- Business process is a service that is an explicitly modeled and executed process consisting of other business services.

Examples are `createStockOrder`, `submitLoanApplication`, and `renewPolicy`.

- **Business Transaction** is a service that changes persistent state of business data or otherwise accomplishes a business action. Examples are `checkOrderAvailability` and `transferFunds`.
- **Business Function** is a service that accesses business function or data without changing the state of the business data. Example is `getStockPrice`.

Now armed with this taxonomy of business services, it helps prospective consumers determine where to first look for reusable services. Or the taxonomy helps architects determine the software stack necessary to address qualities of service. The point is that categorization is often advantageous as the label tells the consumer, the practitioner, something about the service like granularity (such as business sub-process service) or provider type (legacy service). A taxonomy and service classification aids in creating governance models, service ownership models, and reuse as the service repository is organized accordingly. This enables faster access to identifying services that can be reused in future development efforts.

15. Who Owns Service Reuse?

Service reuse is an organizational concern, and ownership should not lie with programmers or developers. Reuse or sharing of services is best achieved when it's understood that the reuse goal is to get the business to reuse more and more business functionalities, both vertically and horizontally. This is facilitated when business stakeholders have a responsibility for increasing sharing and working toward a model where the company can build once and reuse. In many cases, this is also supported if business processes, where it makes sense, can be standard across the enterprise. Service reuse is a shared responsibility between business and IT, and enterprise architecture, having a view to both, should be able to understand and facilitate where service reuse makes sense.

The issue of reuse is getting line of business verticals to share and reuse functionality built and maintained by other verticals and standardizing business processes to promote reuse, which is a governance

issue. Some organizations may need to use incentives to change behavior and to increase collaboration and sharing to promote or realize reuse. Enterprise architecture teams play a huge role in reuse by providing an enterprise view of reusable business processes, rules, services, and information needs required by lines of business. Enterprise architecture teams can use their existing governance roles to help lines of business understand the enterprise-shared service portfolio and assist projects and lines of business with awareness of reusable services and how to consume.

Organizations should adopt a service reuse strategy. Build once and only once is often a guiding principle for organizations seeking to increase reuse. A service reuse strategy should have specific goals, describe organizational responsibilities, and describe activities and tasks necessary to promote reuse. Service reuse strategies may not be the same as software reuse strategies. For example, a software reuse strategy may focus on designing software assets in a generic fashion that allows their use in various contexts. A service reuse strategy focuses on building services with known consumers or known scenarios for consumption. A service reuse strategy is also focused on business reuse, not just IT reuse of services. Software reuse strategies deal with overhead issues of reuse where the overhead may be so great as to favor duplication over reuse. Services reuse strategies deal with the standardization of business rules, information, and processes using reusable services.

The enterprise architecture team, common services organizations if they exist, and the executive or management team responsible for deciding when to buy, modify, or otherwise invest in new IT solutions have the responsibility for reuse. A reuse strategy should define the roles and responsibilities of each of these constituents in the reuse of services.

16. What Are the Common Organizational Pitfalls When Adopting SOA?

The most common organizational pitfall in adopting SOA is the failure to account for organizational change management. The second most common pitfall is to only partially perform organizational change management. As discussed in this chapter, SOA is a major

change for organizations, fraught with obstacles and organizational resistance. Understanding and assessing the culture makes a huge difference to the sustained success of SOA. Everyone may understand the need for organizational change, but months after early successes, the organizational change practices often begin to diminish, communications become infrequent, executive sponsors think that their focus is no longer required, and the change management program loses steam. The partial implementation of change management occurs when one or more aspects of change management are not performed. Whether that be failure to do an assessment of what works and what needs to be improved, a failure to gain consensus from influential stakeholders, or a failure to have measurable goals, all are causes of failure.

Another common pitfall is not looking at the organization knowledge embedded in executives and practitioners who know what works and does not work in their company. Looking at past failures of other good but failed strategies or projects around sharing provides a treasure trove of data on how to avoid such land mines with SOA.

Many organizations underestimate the value of a shared vision and assume that SOA has a common meaning and value proposition. But after hundreds of projects, it is evident that there often is little consensus in organizations as to what SOA is. Documenting an SOA strategy grounded in business needs becomes integral to understanding what changes are needed for breaking down resistance and realizing the benefits of changing approaches and strategies. Lastly, a failure to address governance directly impacts organizational change, because effective SOA governance is required for SOA strategic goals to be achieved.

Organization: Key Concepts

The relationship between business units and IT must evolve if strategic SOA goals are to be achieved. Present approaches, which entrench silos, must give way to approaches that promote sharing. Much of this is addressed with organizational changes as well as changes in applications, governance, architecture, and methods. Business and IT should evolve to a unified state characterized by a

highly collaborative, trust- and business-outcome-focused model. No longer is there a labeling such as “she is from IT” or “he is from business”; instead, parties come together for problem solving, bringing both domain and business expertise to the table.

Organizational change management is a necessity for creating an environment for SOA success. Cultural and behavioral issues surrounding the sharing of services, prioritization, funding, and ownership have to be addressed, and consensus must be reached among lines of business. Barriers to SOA adoption can be flattened, but it requires a holistic approach between business and IT. Service reuse is an organizational concern and focus, with the business playing a huge role in the ownership and promotion of service sharing. Failure to include change management as part of an SOA adoption roadmap is the most common pitfall organizations encounter as an impediment to SOA success.

Content Excerpted from
Pearson Technology Group, Inc.
Not for Resale or Distribution.

Content Excerpted from
Pearson Technology Group by IBM.
Not for Resale or Distribution.

3

Governance

The primary goal of the SOA is to bind the business world with the world of IT in a way that makes both more efficient. SOA is about creating a bridge that facilitates a symbiotic and synergistic relationship between the two that is more powerful and valuable than anything that we've experienced in the past. It is only partly about that bridge—the technology that binds the two worlds; it is much more so about the results that can be achieved from having that bridge in place.

—William A Brown, et al., in *SOA Governance, Achieving and Sustaining Business and IT Agility*

Governance and SOA are regularly discussed in tandem, and it is rare to find anyone who does not admit to the value of governance when adopting SOA. At the same time, reactions to SOA governance range from yawns, cynicism, to enthusiasm. These various reactions result largely because governance is invisible to some, misunderstood by others, and has reached stages of bureaucracy for others. So, is SOA governance a necessity, waste of time, or somewhere in between? This chapter addresses these and other issues through the following questions:

17. What is SOA governance?
18. How does an organization get started with SOA governance?
19. What is the role of change management?
20. Does implementation of SOA tools and infrastructure equate to SOA governance?
21. Should service development be centralized in service centers?

22. Does SOA require centers of excellence, architecture boards, or design boards?
23. Why do organizations need to focus on SOA governance when there is an effective enterprise architecture activity?
24. Is SOA governance required for SOA projects to be successful?
25. How can you measure whether SOA governance is effective?
26. What is the difference between design-time and runtime governance?
27. What are common pitfalls of SOA governance?

Governance: Q&A

17. *What Is SOA Governance?*

SOA governance extends IT governance with the context of SOA. SOA involves people, process and technology, is cross-functional involving lines of business and IT. SOA governance extends all aspects of governance present in organizations necessary for creating specific outcomes (e.g., faster time to market for new products) using SOA. Governance activities focus on the outcomes an organization desires to effect via SOA adoption. SOA governance shines a light or magnifies those aspects of IT governance that should be enhanced when seeking to achieve one or more benefits from SOA adoption.

Figure 3.1 illustrates this concept by depicting SOA governance and highlighting aspects of IT governance that might need to be addressed post SOA adoption. For example, enterprise architecture might establish IT principles, standards, and a common infrastructure, any or all of which might need changes (optimize) after SOA adoption. Such changes might include standardizing on an enterprise service bus and a registry.

Investment processes for provisioning new applications and the prioritization process for those investments are examples of process

changes that might occur upon the adoption of SOA because existing processes would need to accommodate the provisioning and sharing of services, not just applications or systems. The approach currently used by business and IT stakeholders for prioritizing spending for the next calendar year would change as a result of SOA adoption as organizations begin to adopt a shared-services approach in addition to their current practices related to applications.

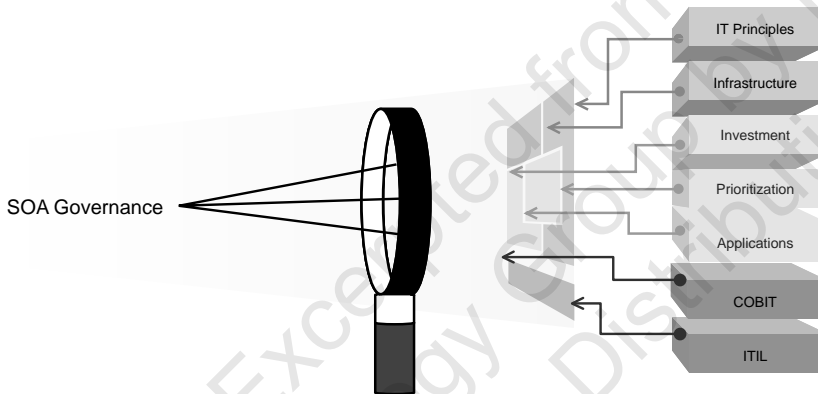


Figure 3.1 SOA governance relationship to IT governance

Organizations that have adopted COBIT (Control Objectives for Information and related Technology), which is an IT governance framework and toolset, or ITIL (Information Technology Infrastructure Library), which is a set of practices for IT service management, might make adjustments to accommodate specific SOA adoption goals in their organization. For example, ITIL provides an excellent list of practices that are largely IT focused. With SOA adoption, it would be useful to have business metrics captured reflecting whether the business process has met key performance indicators, or whether business and IT alignment is progressing according to defined metrics. In this example, the business process metrics and the business/IT alignment, SOA governance activities would surface the ITIL changes to advance SOA adoption.

Numerous studies, books, and articles examine the value of IT governance. Peter Weill and Jeanne W. Ross write in *IT Governance*:

How Top Performers Manage IT Decisions Rights for Superior Results (Harvard Business School Press, 2004) that “effective IT governance is the single most important predictor of value an organization generates from IT.” It is also established that top-performing companies, measured by year-to-year profit and revenue growth, succeed where others fail with the effective implementation of IT governance. Anecdotally, everyone knows the effects of excellent IT governance: Projects get completed on time and deliver the desired business results, costs are lower because infrastructure and applications are shared whenever possible, standards are used to drive efficiency, and excellent relationships develop between business and IT groups. Given the breadth of SOA, it only makes sense that establishing effective IT governance in the context of SOA, organizations can see SOA benefits realized. SOA governance is about changing IT governance to make it more effective using the construct of services and SOA benefits as the change agent.

Different types of governance are present in an enterprise, as illustrated in Figure 3.2. Corporate governance establishes the rules and the manner in which an enterprise conducts business based on its strategy, marketplace, and principles. IT governance defines a structure of relationships and processes to direct and control the enterprise in order to achieve the enterprise’s goals by adding value while balancing risk versus profit over IT and its processes. SOA governance defines the extensions to IT governance to ensure that the concepts and principles for service orientation and its associated architecture are managed and reused appropriately across the enterprise and the stated business goals for SOA and services are met. SOA governance is often a catalyst for improving IT governance.

SOA governance produces the policies, processes, necessary for controlling development, deployment, and management of services. After a service has been deployed, monitoring and management must be instituted to control and supervise the services eco system. Criteria, processes, and policies need to be constantly checked, communicated, and updated. SOA governance activities focus on the service life cycle from inception of service, to monitoring of the service until the service is retired.

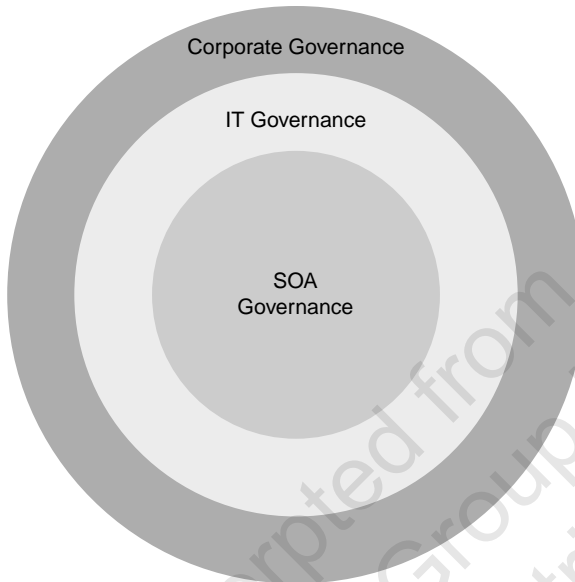


Figure 3.2 Types of enterprise governance

18. How Does an Organization Get Started with SOA Governance?

To initiate SOA governance, an organization must specifically define the SOA goals of the enterprise, line of business, or project. That is, starting SOA governance in a vacuum without the context of goals is possible but often lacks sufficient context to make SOA governance effective. As organizations begin their SOA governance, they must also exercise scope management. After all, organizations don't want to expend undue effort and unnecessary resources to "boil the ocean." Instead, they should "boil a pot of water," define the scope of SOA governance to be commensurate with the context of the initial projects, recognizing that the implementation of SOA governance is gradual and evolutionary.

For example, suppose an SOA initiative is based on these specific goals: increase customer satisfaction, improve time to market, and improve access to information. Suppose further that analysis determined that the existing systems require increased flexibility to realize these goals. Flexibility might be defined as having one interpretation

of customer data versus fifty and exposing a “getCustomerVehicleInformation” to dealers, customers, third-party applications and other applications across the enterprise rather than the present situation where each line of business must sort through fifty different data streams to get accurate information. In addition, root cause analysis demonstrates that new projects would have an increasing need to use this discrete unit of business functionality, “getCustomerVehicleInformation.” Prior approaches of integration are not as efficient as sharing because connecting applications as an approach means spending time in IT development queues, longer testing cycles and this negatively impacts time to market. Prior to SOA adoption, the current approach of integrating applications, contrasted to sharing services resulted, in fifty different interpretations and integration end points for accessing customer data. As a result of this analysis, a decision to adopt SOA is made and the first SOA project will focus on promoting and using shared services, where shared services can be used by multiple applications now and in the future. This decision to adopt SOA is based on looking at the future portfolio of projects, their prioritization, and a determination that this portfolio will need to share discrete units of business functionality. Sharing the business functionality would improve time to market because the service would be built once and reused; customer satisfaction (e.g., the dealers) would be improved not only because of improved time to market but also because of consistent access to information that they need and use.

Based on this scenario, an organization must answer the four questions shown in Figure 3.3 as they are initiating SOA governance. First, the organization must identify the problem it wants to solve. In our example, the organization wants to create an architecture and environment that promotes the sharing of services for a defined portfolio set. Integrating applications is not as efficient as sharing services. Looking ahead at the queue of enhancements and new requirements, there is a clear need to reuse “getCustomerVehicleInformation.” This could be because a new mobile application must be created that needs the service or because a new web based dealer application needs the service. The point is that specific upcoming projects will benefit from using a reusable service, “getCustomerVehicleInformation.” The organization knows that integrating applications creates redundancy, longer testing cycles, and often degrades data quality, so

an assessment of where they are today has been completed. However, this assessment of their current state shows that the organization must also encourage sharing because there is resistance in the organization as everyone is comfortable with the prior integration approaches and reluctant to take on change as they see it as a risk to project schedules. SOA governance is acknowledged as being needed, helping to answer the third question of where do we need to be tomorrow, and a governance model is defined as part of the target state. The governance model describes the people, process and technology differences from the current state. As part of a planning process the organization decided what if any help is needed to accelerate their goals. Many organizations concentrate their limited SOA talent pool and skills in a center of excellence (CoE) to assist projects, socialize SOA thinking, and develop SOA best practices and assets. The SOA CoE focuses on developing skills with people, on push technology adoption, on extending existing processes, and on promoting shared services.

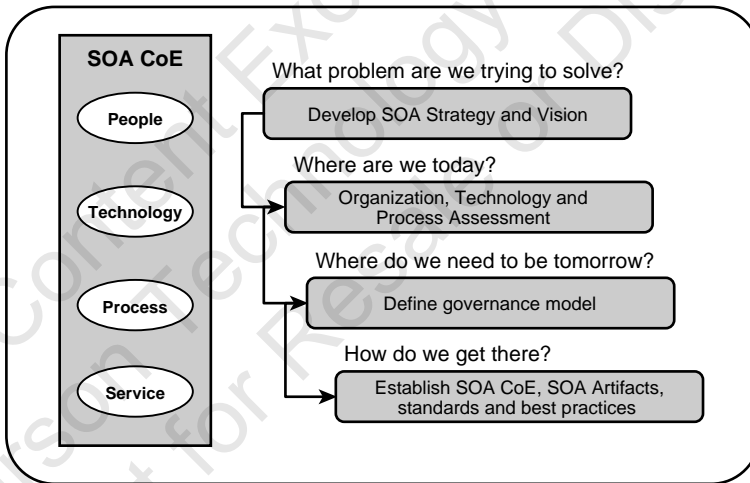


Figure 3.3 How to get started with SOA governance

Getting started with SOA requires a view of where you want to be after SOA adoption is complete. Understanding the problem to be solved defines the scope of SOA governance. The problem could center on cost reduction or time to market savings as examples. However, using the “getCustomerVehicleInformation” example, cost reduction

is achieved by not developing the 51st and 52nd integration points for the mobile application or web based dealer application to access customer vehicle information that incur development and maintenance costs. Time to market is faster because new applications can immediately use the deployed “getCustomerVehicleInformation” service without waiting in IT queues for integration. An SOA strategy would codify this and other examples into a statement about the problem being solved and vision for the future. The organization then determines what is working today and where it needs to be tomorrow to make its strategy work. Based on these considerations, the organization can develop a governance model, a model that defines those aspects of IT governance that must be extended to accommodate the SOA principles and goals articulated in the SOA strategy and vision.

In our example, the strategy is focused on identifying and creating a shared service portfolio using an enterprise service bus (ESB) as the primary technology chosen for SOA adoption; approach for removing connectivity logic from applications; allowing applications to focus on business logic; and, allowing each application to change independently. The ESB would make available shared services for use by multiple consumer applications (e.g., mobile application and web-based dealer application) or other services.

The primary message is this: Organizations just getting started with SOA and SOA governance want to focus on specific problems or goals. The integration example was provided because it is a primary motivation for many SOA adoptions and readily illustrates context for how to get started with SOA governance. When looking at SOA value propositions (such as time to market, cost reduction, or flexibility), it helps to perform root cause analysis: What is preventing the realization of the value today? What must be different to realize the value tomorrow? What actions must be undertaken? In most cases, based on this analysis, organizations recognize the need for SOA adoption and the need to improve, extend, or otherwise modify existing IT governance.

Our experience shows that the process of planning (i.e., defining the SOA vision and the problem to be solved), definition (e.g., the scope of SOA governance and metrics), enablement (e.g., establishing a center of excellence, standards and guidelines), and measuring

(e.g., assessing metrics and evolving the SOA governance scope) makes SOA governance work most effectively. These four activities (plan, define, enable and measure) are illustrated in Figure 3.4. When performing these four activities issues are uncovered which can be addressed as part of a SOA governance activity. Figure 3.4 lists a representative sample of questions that are typically raised.

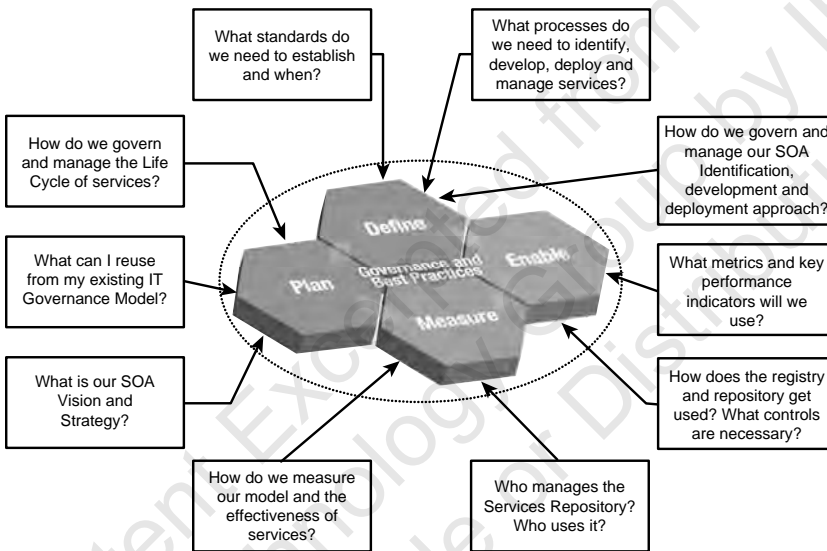


Figure 3.4 Questions for defining the scope of SOA governance

Organizations can develop a laundry list of questions for SOA governance, categorized into buckets of planning, definition, enablement, and measurement. By doing so the scope for SOA governance gets cemented and progress can be plotted against a measurable plan. As a result organizations can avoid the “flexibility bumper stickers” where flexibility is a platitude that appears in strategy documents versus a measurable goal.

19. What Is the Role of Change Management?

Change management is seen from two perspectives in IT. In one perspective, change management is an approach to transform or transition people, groups, or organizations from a current state to a desired

future state. In the other perspective, change management is a process whereby changes to a service are formally introduced and approved before deployment into a next testing stage or production state. In the latter case, change management focuses on both changes to a specific version of a service as it progresses through its development life cycle and changes across various versions of a service that must be managed and governed.

The first view of change management defines it as a structured approach for transitioning organizations from a current state to a future state. In this view, the focus is on deciding what if any organizational changes should be made to realize and sustain SOA benefits. Figure 3.5 shows a fairly typical scenario in which organizations start sharing services between lines of business. Everyone agrees that the optimal solution is shared services. However, questions arise concerning who funds the shared service, who owns the shared service, what the responsibilities of a service owner are, and what a service owner is. In some cases, the project stalls as a result of these issues and churn, resulting in missed expectations or project failure. Chapter 2, “Organization,” addresses how to use change management to resolve issues on funding and sharing so as to avoid missed opportunities and expectations.

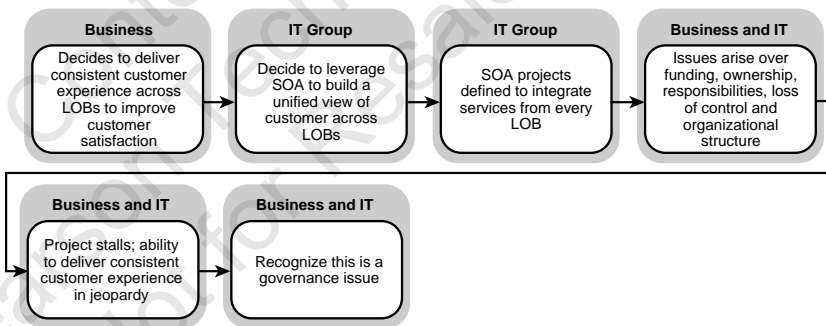


Figure 3.5 Role of change management for SOA governance

Using our example of integration and the ESB implementation in the preceding question, change management is necessary to get lines of business to not only share services but also to share the common infrastructure, the ESB. This results in sustainable cost reduction and promotes sharing of business and technology aspects of

SOA. Change management involves communication, organizational design and change, as well as SOA governance so that executives and managers are aware to what they have committed to in SOA governance. Lack of clarity of roles and responsibilities makes it difficult for leaders to influence and educate their teams. Without visible key stakeholders, support for the adoption of SOA governance will dwindle; and the value propositions for SOA will fail to materialize. Change management is essential for successful and effective SOA governance.

The other perspective of change management focuses on changes to a specific version of a service or changes across versions of services (which must also be governed and managed). IT governance for most organizations addresses versioning of software artifacts and release management. If current practices are deficient, they will also be insufficient for SOA. SOA governance and change management play a role in helping organizations deal with the issues of deprecation and staging of services through test stages and ultimately production deployment. Changes to the service interfaces, service implementation, and service contracts must be governed and managed. The role of change management is closely tied to service versioning and is responsible for managing proposed changes to the service portfolio so that the ripple effect and impact of change is contained and minimized to the extent feasible. Services must not only perform consistent with the service contract; services must also be discoverable, the contract interface understandable, and of course, the service must adhere to a demonstrable and proven set of test cases. The service must be stable.

20. Does Implementation of SOA Tools and Infrastructure Equate to SOA Governance?

The software industry has responded to the need for SOA governance by providing software products that support SOA governance, most notably registries and repositories. Such tools provide information about services—metadata that supports versioning, discovery, and management of services both at design time and runtime. It is a well-established best practice in the industry to establish the processes first and then do tool shopping. Establishing the process first and using it provides organizations an opportunity to see what works, what tweaks are required and where gaps exist. The process definition becomes an

input feed in helping to define the tool requirements. Of course, the process and tool feed off each other as the process gets modified to reflect what is possible or optimal to perform using the tool. Tools and technology alone does not equate to establishing SOA governance. SOA governance as described in this chapter requires a range of activities (e.g., organizational change, metrics, updated processes and technology adoption) for SOA governance to be established and effective. Design-time and run-time governance are examples of where both process and tools are required for SOA governance. In both cases, understanding the design-time and run-time life cycle process would benefit the activity of selecting the optimal tools.

Design-time governance entails the activities centered on application development using services. Design-time governance covers the full system development life cycle, including requirements management, architecture, design, development, test, documentation, and production deployment. Design-time governance is necessary because it focuses on making information about a service (service descriptions) available at the right time. Design-time governance addresses change management in the context of versioning and release management.

Runtime governance addresses the execution and operational aspects of a service. Monitor the service in the context of business transactions or business activities so that the business can be informed of bottlenecks or other impacts to key business transactions or activities. Effective runtime governance detects performance bottlenecks (for example, throughput or availability) before they occur. Run time governance may include business activity monitoring so that metrics about the performance of business processes (e.g., it completed successfully in a certain time interval). Run time governance can work with policies. Runtime governance should monitor all aspects of service execution with a transaction or business process context.

Various software products can support and automate aspects of design-time and run-time governance, but the tools or software products alone is not sufficient for governance. The effectiveness of design-time and run-time governance is accomplished by having effective processes, people, and the right tools.

21. Should Service Development Be Centralized in Service Centers?

Some organizations developing services have centralized the service development process by creating service centers or centralized service development groups. Often, this approach is selected as an initial way to leverage a limited talent pool available for service development and to ensure that service development uses a consistent set of standards, tools, or architecture building blocks. Centralized service development can provide superior control and enforcement of SOA standards, but on the downside it can operate as a slow funnel for rapid development of services.

Organizations have at their disposal a variety of organizational structures to support service and application development. Factory models and centralized delivery models can all work successfully, but at the same time project management accountability must be present to minimize finger-pointing when, for example, consumers and providers blame each other for schedule slippage or other delivery issues. Centralized service development often means the service provider role will be centralized and service consumers will provide requirements about their service needs. This scenario can create the undesired effect of waterfall development, where the service consumer requires all requirements be well formed in advance of any service development.

Services that have an enterprise scope are often excellent candidates for centralized service development or service centers. Services that require domain knowledge from a line of business and need to be shared by other lines of business are often good candidates for line-of-business service development versus a centralized model. Determining whether something is an enterprise-level service can often be done by looking at whether the business process spans the enterprise or is specific to a line of business.

Effective SOA governance is a critical success factor when organizations employ factories or services centers for service development. Design authorities can be used to direct the build and construction activities. Using design authorities allows projects to be reviewed for architecture and design compliance, providing the necessary controls for effective SOA governance.

22. *Does SOA Require Centers of Excellence, Architecture Boards, or Design Boards?*

A successful SOA program that helps transform the enterprise into an agile one with supporting adaptive IT infrastructure requires a combination of enablers. One key enabler is the implementation of governance around SOA. This governance is not achievable without a governing body consisting of respected technical and business leaders within the organization who collaborate to achieve consensus on architectural and design aspects that impact the organization as a whole.

An SOA CoE (Center of Excellence) facilitates the realization of business value through the implementation of SOA and leads corporate-wide business and technical communities in enabling business agility through shared and reusable services. The SOA CoE crosses operational and organizational boundaries to enhance awareness of shared services, operates as a technical aid to projects, provides education and training to projects teams, conducts architecture reviews, promotes asset adoption, resolves technical issues, and provides instructional guidance in the context of active projects on SOA standards and best practices.

An architectural board is often an existing enterprise architecture board or a newly established SOA leadership board that provides visibility and commitment to SOA adoptions. Often, the architecture board performs four governance processes: compliance, exception and appeals, vitality, and communication.

Governance Process	Process Description
Compliance Process	Provides the review, approval, and reject process using criteria agreed upon as part of establishing the governance model.
Exceptions and Appeals Process	Describes the process for allowing appeals and exceptions for non-compliance to standards, architecture, or other guiding principles.
Vitality Process	Process for keeping governance applicable and pertinent to all stakeholders. This process defines the activities that must consistently be performed to sustain a collaborative and healthy governance program.
Communication Process	Defines the process for continuous education, training, and communication of different stakeholders necessary for the sustained success of governance.

Design boards or design authorities may be the same as an architectural board and perform similar functions, or organizations might assign different responsibilities to each. Design authorities participate in quality-assurance reviews for shared services, identify services that can be reused by project teams, and direct project design activities that relate to shared services.

These boards are formal constructs that require participation and support from all business lines. Such boards will require time and patience to become productive as it takes time to mature the relationships and create fruitful consensus-driven collaboration; to agree and adhere to a set of guiding and common principles. Therefore, it is essential to grow boards as organizational capabilities, which in some cases might mean “bootstrapping” or growing gradually with increasing participation and circumference of influence.

Figure 3.6 shows the relationships between various organizational constructs such as design authorities, architecture boards, and CoEs. The SOA CoE works directly with projects to flatten SOA issues and to accelerate adoption of SOA. The SOA CoE may perform a variety of activities to fulfill its mission, which is ultimately to make projects teams more efficient and effective. These activities may be on-the-job training, skills transfer, subject matter expertise, development assistance, or anything required making the project team effective. SOA CoE team members in some organizations will take direction from an architecture board or design authority. The architecture board or design authority provides project direction and may escalate to the SOA leadership board for approval of important decisions. The architecture board or design authority is also responsible for the architecture and establishes architectural direction for SOA to application development teams in the use of architectural frameworks, standards or reference architectures.

The SOA leadership board provides the tactical leadership needed to direct and control SOA activities across the enterprise, which includes establishing SOA governance policies, standards, and processes. This board ensures the vitality of SOA and that necessary communication occurs among all stakeholders across the enterprise. This board tracks and reports against defined metrics to gauge the progress of SOA in achieving its strategic and tactical benefits. An

SOA executive steering committee provides visibility and commitment to SOA within the enterprise and brings proper focus to bear when necessary to remove political roadblocks or other obstructions. The SOA executive steering committee also ensures business involvement and commitment to SOA adoption.

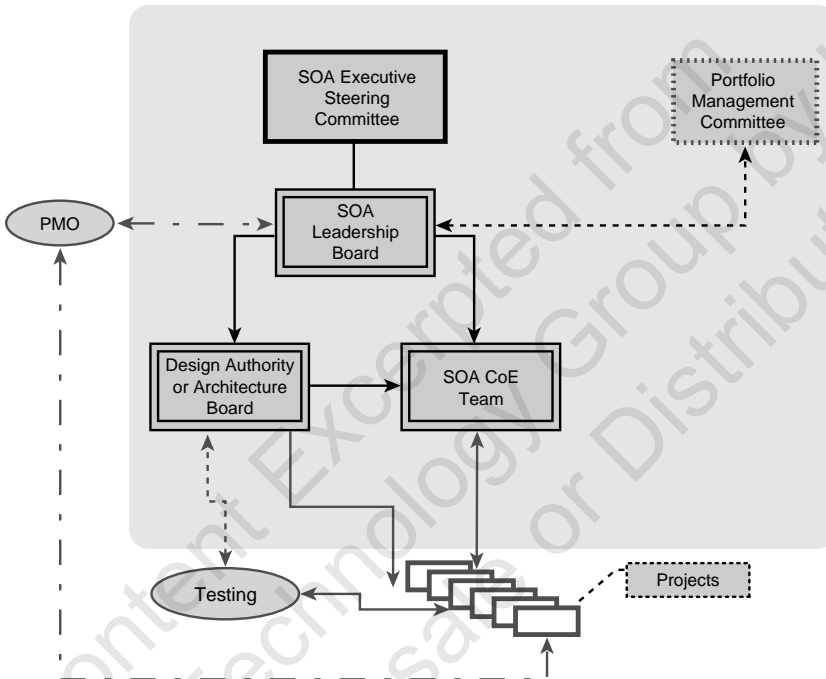


Figure 3.6 SOA CoEs and architecture boards

The project management office (PMO), if necessary and in place, works with the various committees, boards, and CoE to ensure that projects are delivered on time, within budget, and consistent with measurable SOA goals. For example, if flexibility is a business goal, all parties depicted in Figure 3.6 will work cooperatively to ensure flexibility is not just a platitude but also a measurable feature to be realized in one or more projects. Most organizations have a function responsible for portfolio management where projects are funded and prioritized.

23. Why Do Organizations Need to Focus on SOA Governance When There Is an Effective Enterprise Architecture Activity?

SOA requires additions to standard enterprise architecture practices and processes. The notion of a central service model, for example, is unique to SOA; enterprise architecture must adjust to address handling of service models as a vehicle to promote sharing across the enterprise. Governance plays a role in how enterprise architecture (EA) teams augment their current practices to address SOA. Organizations with effective enterprise architecture perform four governance processes and they use these same processes with the adoption of SOA:

- A **vitality process** maintains the applicability and currency of the architecture reflecting the business and IT direction and strategy. Architectural principles are often used to guide the vitality process. SOA adoption adds new architectural principles such as, “Service models will be used to capture the enterprise portfolio of shared services.”
- A **compliance process** reviews and approves or rejects the design of a solution. This process can be performed at various points throughout the business and project life cycle. EA teams will review SOA artifacts for compliance to SOA reference architecture or standards as an example.
- A **communication process** educates and communicates the architecture across the organization. This includes ensuring easy access to and consumption of architectural information and assets. Implementing SOA requires communication decks, white papers and training materials be updated to reflect SOA adoption at the enterprise level. This includes standards, architectural guidance, reference architecture and refinements to any architectural building blocks necessary for SOA.
- An **exception and appeals process** allows projects to appeal the noncompliance of a solution or design decision or investment with the board and perhaps be granted an exception. Project teams will need to make architectural decisions that in some cases may conflict with an architectural standard. The EA team will listen and grant exceptions as necessary for projects employing SOA.

SOA governance uses the exact same governance processes found with effective enterprise architecture. Where effective IT governance is in place, SOA governance operates solely to enhance. SOA governance does not introduce any new governance processes; it introduces new processes to be governed, such as service identification, service design, service funding, service domain owners, and service runtime.

The following table reflects the influence EA has on SOA projects at the enterprise and project level using models and guidance. An example of a model at the enterprise scale is an architectural framework that organizes architectural building blocks. Design frameworks for SOA accompanied with code is an example of a model that can be used at the project scale. An example of guidance is a reference architecture that could be used across the enterprise or at the project level providing prescriptive guidance on how to elaborate a solution's SOA.

	Models	Guidance
Enterprise scale, or things which help plan and organize work	Enterprise Models	Architecture building blocks, Usage principles, Reference models
Project scale, or those things focused on building or implementing things	Program and Project Models	

A focus on SOA governance often provides the genesis for updating EA models such as an architectural framework. Figure 3.7 depicts a workflow of various EA activities culminating in using the architecture framework that has been updated to address SOA. The architecture framework can be automated and visually represented where project teams, solution designers, can programmers can find and reuse content. EA architectural frameworks can also be linked to repositories to facilitate locating services at design time.

Figure 3.7 illustrates a workflow where services, design patterns and reference architecture are architectural building blocks reflected in EA model, enterprise architecture framework, an enterprise scale model, to facilitate classification and location of architectural building

blocks. In this workflow the EA provides a framework to facilitate reuse, assets to providing project guidance and assets that help in the construction of applications. Solution designers use the architectural framework to accelerate development of applications or services. For many organizations, it will be the output of activities associated with SOA governance that causes the enterprise architecture framework to get updated to reflect SOA building blocks. In some cases, SOA governance will be the genesis for the renewal and a more effective Enterprise Architecture.

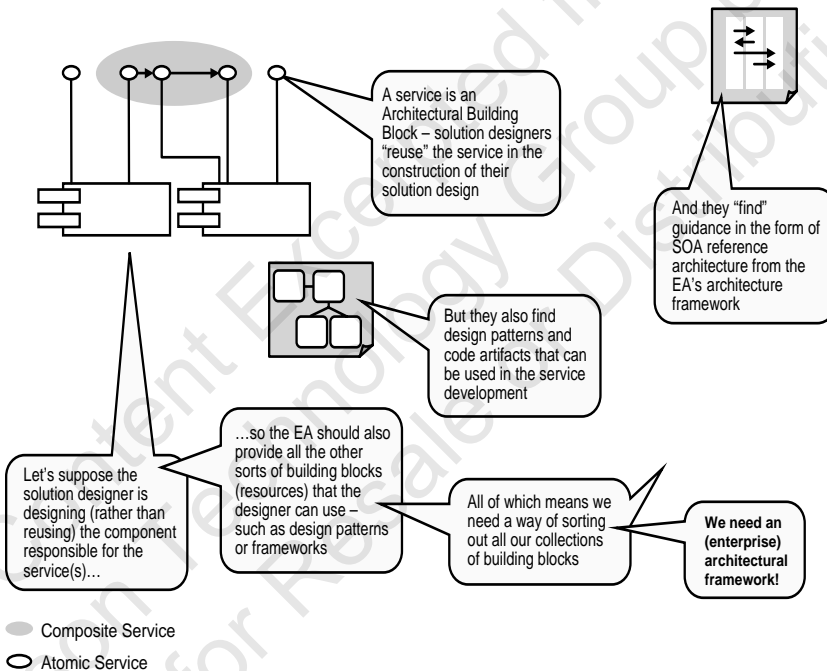


Figure 3.7 Enterprise Architecture framework

24. Is SOA Governance Required for SOA Projects to Be Successful?

Organizations can successfully implement an SOA project without SOA governance; however, the strategic and tactical benefits of SOA cannot be delivered without effective SOA governance.

Ensuring that SOA projects produce acceptable results requires key performance indicators or success metrics that provides the expected results from SOA governance. Thus, one of the processes to be governed is the SOA system development process providing guidance on how to identify, design and develop reusable services. Such guidance avoids service proliferation where a lot of services are developed but very few are reusable. SOA governance focuses on people, processes and technology that moved in unison to increase the reuse of services.

When SOA is adopted, we find organizations at different levels of maturity in IT governance and SOA governance. In some cases, this means that each line of business looks after its own interests, and the result can be inefficiency, higher costs, and a viewpoint that it is always cheaper for a line of business to develop something itself rather than reuse a service or infrastructure built by another line of business or organization. Most organizations have good citizens, so often the reason for a lack of commitment, to a shared or centralized strategy is a perception of higher risk (i.e., “do not have control over my destiny because I must rely on other organizations”) or its cheaper or faster to do it within the line of business than to work with other lines of business or a centralized team. Effective SOA governance addresses the people, process and technology issues that may prevent achievement of both strategic and tactical goals for SOA adoption.

25. How Can You Measure Whether SOA Governance Is Effective?

SOA governance requires measurements in several areas: business benefits, project costs, and service utilization. Potential business benefits and associated metrics can be tracked to determine whether business benefits are realized from SOA adoption. If a desired business benefit does not accrue in the planned time horizon, it is a sure bet that there is ineffective SOA governance. Effective SOA governance also helps organizations identify the likelihood of a benefit accruing and when.

SOA governance has a number of key processes that should be implemented. One of these is a process that ensures vitality and currency of the SOA governance policies and processes. Metrics are gathered at key points to provide feedback for ascertaining that the

governance processes in place are indeed effective. Defining metrics at key stages of the service life cycle and system development life cycle is a most effective way of gauging the effectiveness of SOA governance.

For most organizations, when there is a failure to achieve any of the following, a red flag should be raised as to the effectiveness of SOA governance:

- Reduce the time to deploy business functions or changes to existing functions
- Reuse SOA assets by other projects or lines of business
- Improve flexibility of applications
- Utilize ESB to reduce costs
- Reduce maintenance costs
- Achieve development savings in new development of shared services

26. What Is the Difference Between Design-Time and Runtime Governance?

Design time governance includes the definition of policies and proper life cycle associated with a service as it is designed, tested, implemented, monitored, and registered in the service registry. Design-time governance provides a full life cycle view of a service from inception to deployment to retirement. Design-time governance uses registry and repository tools to track service design, management, policies, and any artifacts associated with the service. Such artifacts might include a test report demonstrating that a service successfully passed certain quality-assurance tests. Design-time governance includes design tools to facilitate the modeling and creation of services, deployment tools addressing service implementation, and test tools.

Runtime governance uses the operational policies to monitor the runtime execution of the services against the policy criteria defined and against operational requirements such as service level agreements. Runtime governance practices address managing the quality of a service such that a service is known in the context of its application flow or business transactions. For example, services from some

categories of users may allow a 5-7 second response time but for others it must be 2-3 seconds. Or at a certain time of day, inquiry transactions receive a lower priority than deposit transactions. Service dependencies and consumers, across a heterogeneous environment, are known as part of runtime governance allowing context rich policies to be defined. Metadata is accessible at runtime and describes the expected service availability, throughput, business owner, and any other pertinent information necessary to manage, secure, and operate the service. Security is another aspect of runtime governance where enforcement occurs through authentication, authorization, or credential mapping. Service virtualization to handle load balancing, routing, or failover is another aspect of runtime governance.

27. What Are Common Pitfalls of SOA Governance?

“Trying to do too much too soon” is a common syndrome. “Iron-fisted” or heavyweight governance is another common problem. In addition to these two common problems, the following are common pitfalls of SOA governance:

- SOA governance becomes solely about a focus on integration.
- Lines of business resist SOA adoption and avoid sharing services.
- SOA efforts become “shelf ware.”
- Funding issues and SOA projects drag to a halt.
- Failure to achieve reuse or time to market savings.
- Toothless governance.
- SOA goals take a backseat to tactical project goals.
- SOA governance is ineffective if poor IT governance prevails.
- Selling SOA versus specific measurable strategic or tactical benefits of adopting SOA.

SOA is more than the sum of the technologies that enable SOA. SOA governance recognizes the synergistic and overlaps between technology, people and processes to achieve strategic SOA goals. SOA governance should have a scope focused on achieving one or more measurable strategic or tactical goals. SOA governance requires active support from senior executives with authority and influence. With large-scale projects, SOA is often a strategic goal with a larger

focus on delivering a business solution with its own sets of challenges. Senior executives can ensure that the strategic goals are baked into the project scope. Architects can make sure these goals are measurable and achievable.

Defining value propositions for lines of business to use and share services and SOA assets is something that SOA governance and change management can positively influence. Management support and delegation of the SOA charge to key respected leaders coupled with grassroots building of consensus of the method to gradually deploy governance is essential.

Establishing an SOA funding model for both short- and long-term initiatives is another critical success factor for SOA governance. There must be some commitment to funding dedicated resources necessary for SOA stewardship (e.g., SOA CoE and SOA governance activities) and the procurement of supporting SOA tools (e.g., design time and runtime) and technologies. Funding for SOA projects must be crafted in a manner that advantages and incents lines of business. Establishing incentives that reward lines of business for serving enterprise goals is an example. Governance bootstrapping that starts with a lightweight approach to governance consisting of an architecture review board consisting of key respected leaders from both the business and technical sides is a prudent and effective way to get started.

Governance: Key Concepts

Governance is about establishing chains of responsibility, authority, and communication to empower people (decision rights). Effective governance requires establishing measurements, policy, and control mechanisms to enable people to carry out their roles and responsibilities. Governance determines who is responsible for making the decisions, and management is the process of making and implementing the decisions. SOA governance often entails the reengineering of IT governance as SOA governance shines a light on IT governance, in much the same way that data governance did to IT governance some decades ago. We talk about SOA governance as separate from IT governance not because it is separate but because of

the focus on what improvements or changes are required to existing IT governance for SOA benefits to be realized and sustained.

SOA governance is not a one-size-fits-all solution, and the scope of SOA governance and defined metrics are key success factors for effective SOA governance. SOA governance matters if organizations are to realize the business benefits of SOA (e.g., business flexibility and improved time to market savings). SOA governance mitigates business risk, helps maintain the quality of services, and ensures consistency of services. SOA governance also improves team effectiveness as we measure the correct things, and communication between business and IT is improved.

Real-world experience demonstrates that effective governance coupled with a compelling SOA vision and a proactive plan provides big payoffs for organizations. Governance is not just about compliance; it is about promoting the right projects and making them better. With the right focus, support, and funding, SOA governance can be an enabler by facilitating reuse, prioritizing spending, reducing costs, and setting the technology direction. There must be a concerted effort to streamline and empower governance processes wherever possible, giving them teeth and making them efficient. Centers of excellence provide an opportunity to significantly accelerate an organizational path up the SOA learning curve and actually bring the focus to business impact and innovation. SOA CoEs also provide employees with an enabling environment for expanding skills and advancing their careers.

4

Information

Information architect: The individual who organizes the patterns in data, making the complex clear; a person who creates the structure or map of information which allows others to find their personal path to knowledge; the emerging 21st century professional occupation addressing the needs of the age focused upon clarity, human understanding and the science of the organization of information.

—Richard Saul Wurman

A significant dichotomy has always existed between the worlds of processing and information and between the dynamic and static aspects of software engineering. Although processing is not fully dynamic and information is not fully static, information pertains to those aspects of the domain that remain constant over time (i.e., the business entities that remain persistent throughout the processing part of an application). Customer information may get updated or transactions posted to an account or ledger, but there is a business entity, account, or customer that is constant. Information plays a central role in information technology. Recall that IT in the past was referred to as data processing because the heart of IT is data and processing of that data.

With the advent of SOA, information is now available as a service. Information is passed to a service through the input message arguments and processed and persisted in the back-end systems. The results are passed back to the service consumer who originally invoked the service operation via a message. Access to a heterogeneous information environment can be sanitized via an information façade that hides the complexity underneath and uses a canonical

data model to provide guidance for message schema used by services. Information is a cross-cutting concern that touches the user interface design, workflow, business process design, and the service design. Organizations would like to have information available anywhere at anytime to authorized parties, and SOA enables this goal using information as a service.

In this chapter, we explore questions related to the intersection points between information architecture and SOA. The convergence of information architecture and SOA improves the reuse and flexibility of services. The following questions are asked and answered:

28. What is the relationship between information architecture and SOA?
29. What are information services?
30. How are information services classified?
31. Do information services differ from other services?
32. How should information services be identified?
33. When should information services perform create, read, update, and delete operations?
34. Are information models required for effective SOA implementations?
35. What is a canonical message model?
36. How should a canonical message model be created?
37. Can SOA improve data quality?
38. What are the common pitfalls with information architecture and SOA?

Information: Q&A

28. What Is the Relationship Between Information Architecture and SOA?

Information plays a central role in the message design and flow between services. The information architecture relationship with

SOA is to provide a common structure and meaning for data shared across SOA layers and business domains for all parties in the ecosystem—consumers and providers. It provides a common vocabulary that controls the common definition of terms and facilitates the reuse of services. Without an agreement of terms (e.g., what is a customer, account, address, or name), it is difficult to implement services related to those terms. Both business and IT stakeholders should have a common understanding of business terms, which the information architecture provides. This helps with the proper definitions and corresponding structures to define the inputs and outputs of a service, its messages.

Service messages are more complex than single data types. Messages represent entities and their relationships. Leveraging information architecture data models enhances the design of messages. Aligning the service and data models accelerates design and avoids unnecessary transformations of data between applications or services.

However, this is no guarantee that the quality of the data being returned by services will be accurate. Data, which meets the rules and constraints of its original repository and application, may not satisfy requirements on an enterprise level for a service. Data quality issues might not become apparent within the original application but may cause problems when exposed more broadly on an enterprise level with SOA adoption. Missing values, redundant data types, and inconsistent data formats are often obscured in applications and become problematic when exposed to new consumers in an SOA. The information architecture is necessary for effective and efficient message design of services in SOA adoptions.

29. What Are Information Services?

There is a category of services for which the creation uses information architecture and where a separation of information from applications and processes occurs. This type of service is referred to as an information service. Information services are a specific type of service that encapsulate the underlying information entities and their data sources. Information services can provide processing, consolidation, partitioning, cleansing, validation, and transformations necessary

to fulfill a request to access, update, create, search, or validate information or data. Information services consolidate underlying data entities, accessing multiple and disparate information sources, transforming and consolidating the results into a format acceptable to the requesting party, the consumer.

Information services are often used to address the heterogeneous nature of data sources and the fact that data sources often are replicated across several vertical systems. When presented with these two challenges, information services can be used to eliminate inconsistencies in business processes. Reusable, enterprise information can be viewed as sets of business entities standardized for reuse across the enterprise and used to create standard structures, semantics, and service contracts. The goal is to create a set of services that become the authoritative, unique, and consistent way to access the information.

30. How Are Information Services Classified?

Information services allow the consumer to retrieve information in a variety of formats using generic interfaces that increase the reusability of the service across heterogeneous platforms and vertical systems. Accomplishing this goal generally involves classifying information services as follows:

- Integration services that are responsible for data cleansing, data transformations, data consolidation, or federating data across multiple data sources to provide a consistent and authoritative data source. Integration services provide a service consumer access to consistent and integrated data that resides in heterogeneous sources. Integration services are most often read-only.
- Data services handle queries and the typical *create, read, update, and delete* (CRUD) functions. Data services access structured data as a service.
- Content services expose federated content, imaging data, archival records, or record management. Content services manage distributed and heterogeneous unstructured information so that a service consumer can access the content seamlessly.

- Master data services manage and expose trusted master data as services. Master data services provide accurate, consistent, and contextual access to master data from data residing in heterogeneous and inconsistent sources.
- Analytical services provide insights as data is sourced from demographic data stores, merchandise, contacts, transactions from data warehouses, to create analytical information. Analytical services provide access to analytic data out of raw heterogeneous structured and unstructured data. Analytical services are mostly read-only.

Each of these information services can be derived as follows:

- Implementation via a direct access to one or more databases where mapping of the service interface to the physical data schema is the only requirement to identify data elements for input and output messages. However, business rules might exist for addressing the integration of data from multiple data sources.
- Implementation using a preexisting *application program interface* (API) typically used in one or more applications to get at one or more data sources where the data access is only allowed using an application. In this case, the data exposed on the service interface is derived both from the application API wrapper and the underlying data sources. Using only the API may not be optimal because it might not satisfy the needs of the intended consumers of the service, and therefore, analyzing both the API and underlying data to determine the data elements for the service interface yields a more reusable service.

Figure 4.1 illustrates the fact that information services are part of the Services Layer and are access applications, databases, or service components. Like all services described in the Services Layer, the service is realized using components or applications. Information services can be atomic (i.e., single service) or composite services (i.e., aggregates other services) and can leverage multiple applications/databases to achieve their functionality.

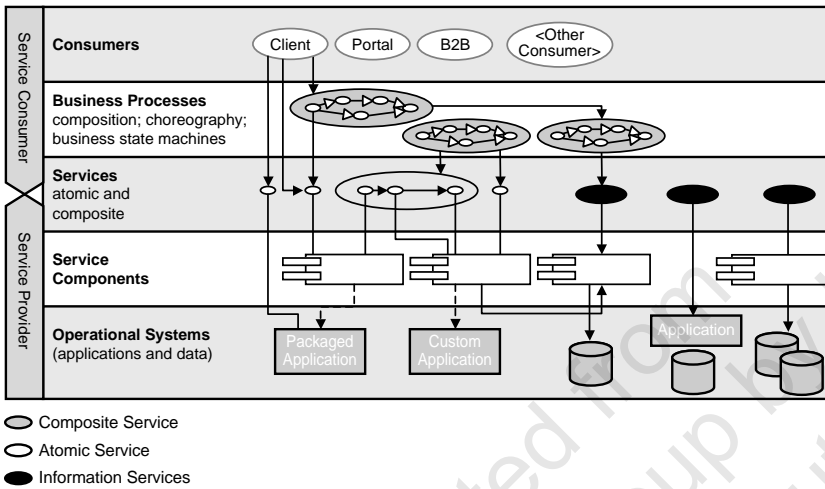


Figure 4.1 Information services

31. Do Information Services Differ from Other Services?

Information services are a type of service, and like all services, an information service can be a reusable unit of business capability or functionality. With SOA, the word *service* has a specific context; that is, services are reusable and participate in and are composed in a value-net, enterprise, or line of business to fulfill business needs. Understanding the relationship between services and service components can help you understand the relationship between information services and other services in an SOA.

Services provide the formal contracts between the Consumers Layer and the Providers Layer. The Services Layer provides the mapping from the business process to the service implementation. The Services Layer is responsible for identifying the correct service provider for the request from the consumer, locating the service implementation, binding to the service implementation, and invoking the requested service operation.

Service components provide the implementation layer for services. The Services Layer exposes interfaces from the Service Components Layer. There is a many-to-many relationship between service interfaces and service components. One service component may be

exposed into different formats by different service interfaces. Multiple service components can be combined in the Service Components Layer and exposed in a single service interface. Composite services can be built in the Services Layer to combine multiple existing published services creating a different service. Service components can be clustered into subsystems (e.g., loan processing or order management). Service components can be decomposed further into reusable composite parts comprising the component implementation.

Figure 4.2 highlights the difference between a service style interface versus an API-like style interface. Figure 4.2 also illustrates an information service that, in this case, is a request for mortgage information. Unlike other services, information services are focused on returning information that often represents an aggregation of multiple data sources or a snippet of information previously available only in business intelligence (BI) applications. Prior to information services requests for data stored in data warehouses, data marts, or other reporting systems, would require accessing those applications. Now with information services, business intelligence data becomes easily retrieved without the weight of having to use a BI application.










By Post – the Service Style	By Phone – the API-like-style
<ol style="list-style-type: none"> 1. Client requests mortgage information 2. Provider sends it 3. Client uses mortgage information in a variety of contexts of its own choosing 	<p> Client calls provider</p> <p> Provider asks "Hello, how can I help?"</p> <p> Client: "I'd like some information on my mortgage please."</p> <p> Provider: "What is your name?"</p> <p> Client: "Bond, James Bond "</p> <p> Provider: "What is your address?"</p> <p> Client: ...</p> <p> ...</p> <p> Provider: "Ok, your mortgage agreement number is 12345;</p>

Figure 4.2 Service interface versus API

Service components can be entirely custom coded or service components can be a way of wrapping or connecting to existing components or subsystems in existing operational systems including BI system whether they be Teradata, Cognos, Informatic, SAS, or others. A Service component is an approach for making information services available, as the service component is the connection to existing systems (e.g., BI systems), or it becomes an approach to wrapping legacy systems. In either case, an information service can be exposed.

An information service, based on its functionality, can be the same as a business service. For example, `retrieveMortgageData` could be a legitimate business service that also happens to be an information service. Information services can be composed in other services. For example, when a business process invokes a service called `submitOrder`, the implementation can be a collaboration between a business service inserting the new order into the Orders system and an information service providing the geospatial analysis data required to allocate order fulfillment to the distribution center nearest the delivery address for the order.

32. How Should Information Services Be Identified?

There are two primary reasons why practitioners should identify services early in the development cycle instead of making service identification ad hoc—that, is a bi-product of component or object specification. Identifying services early allows for services to be used for structuring the application such that objects and components are identified in the domain of the service. This allows the service to enforce structuring, and it means the service has two aspects: a business area focus and an IT focus later for component boundaries and interface definition. The second reason for early service identification is it increases the opportunity for reuse as the separation of concerns enforced by the service decouples it from other services, making it easier to understand how it can be consumed by processes or applications. For these same reasons, information services should also be identified early in the life cycle.

Answering the following questions can help identify information services:

- Is there a need for data cleansing or data standardization?
- Does the solution require retrieval or maintenance of data without applying any business rules or behavior?
- Is there a requirement to return (read-only) results of a complex data analysis such as hidden patterns, summarization, predictions, relationships, or trends?
- Is there a need for a reusable function to retrieve or update data from one or more data sources for which no existing application exists?
- Is there a requirement to retrieve data from heterogeneous data sources or operational systems?
- Is there a need for data access security so that the existence of an information service allows usage of various SOA security policies to be applied to those services?

Exploring any of these questions and their answers leads to identification of information services. In the same way that not every access to business logic from a consumer goes through a service interface, not every access to data should go through a service interface. Therefore, not every API or every data access should be exposed as a service.

33. When Should Information Services Perform Create, Read, Update, and Delete (CRUD) Operations?

In the early days of SOA, there was a lot of service proliferation, and in some cases information services that were created performed only *create, read, update, and delete* (CRUD) operations. A service should not provide generic access to data such as “give me any information that I need in any format that I define” or simply be a replacement for using a query language like SQL. A query language has greater flexibility for various types of queries and CRUD actions. In most cases, services that are nearly CRUD operations should be augmented by business logic that provides a set of rules for accessing,

manipulating, converting, validating, and maintaining the consistency and integrity of the underlying information. CRUD services alone do not provide added value and represent a poor choice of service exposure unless some additional value is provided. There are occasions when information services should be created that largely perform CRUD operations. One example is where fine-grained data access security is a concern, and a separate service allows that security to be more easily performed using security policies. Another, of course, is for any of the reasons cited in the previous question.

34. Are Enterprise Information Models Required for Effective SOA Implementations?

Enterprise information modeling is typically a corporate activity that produces models of the information resources in an enterprise. An enterprise information model is a rationalized set of business entities and attributes that capture the primary abstractions of the business at a conceptual level. It drives the vocabulary and semantics for the enterprise or line of business. Information models by design provide documentation for the concepts being described, and they facilitate the reuse of services and data. So, having enterprise information models makes any architecture that uses such models more effective, including SOA implementations.

In many instances, actual information models exist in silos and in different business units. Different divisions within lines of business may have different information models or data models. Organizations often have many models available, with each describing a portion of the enterprise and each having been developed independently. The quest for a unified and rationalized enterprise-wide information or data model has been one of the holy grails of information architecture. Seldom have organizations been successful at creating enterprisewide information models or data models that are adopted by all lines of business. So many sound reasons exist for creating smaller or line-of-business models as they are easier to construct. In the absence of an enterprise scale data model, models focused on a smaller domain or lines of business models are useful for identifying information services.

Although the physical consolidation and rationalization of an enterprise information model may be an unfeasible endeavor, the creation of a high-level enterprise lexicon of business entities that allows different lines of business to speak abstractly about the fundamental notions of their business is, in fact, a pragmatic and useful practice. Therefore, an enterprise information model can be utilized not for the physical database level of manifestation but for rallying a common understanding of terms across lines of business and divisions within an enterprise. This promotes reuse, which is a key benefit and utility of an enterprise information model. For example, variations of an account entity may have different attributes in different lines of business. However, each line of business can agree on what is meant by an account even though the data attributes may diverge or change over time.

The invocation of a service to process loans, for example, may require a different set of attributes for different business lines (e.g., whether wholesale, retail, or government). However, the fundamental principles of information hiding and process hiding can be applied and combined with information as a service. In this way, a uniform piece of functionality is developed, invoked, governed, and managed across the organization. The underlying implementations will be redirected based on the context in which the invocation is requested. A lookup to a service registry can be made and policies identified and routed to the appropriate service, at run-time, for a given business line. In the same way, the underlying data structures for a particular line of business are discovered through the use of policies and rules. When the context is passed through to a business service for loan processing, it can use information as a service to locate the appropriate information sources to transform and convert data to the suitable format, mapping source to target formats.

Information models and data models are both abstract models, where the information model is not a type of data model, but one that provides representation of business entities their properties, relationships, and operations that can be performed on the entities. This is in contrast to a data model, which describes how data is represented, relationships, and how data is accessed. Service message design benefits from information models and data models. The next answer

about canonical models provides a richer explanation about the relationship between these abstract models and SOA.

35. What Is a Canonical Message Model?

The canonical message model is a standardized format in an enterprise or line of business for exchanging information. The model provides the default business data interchange so that services and their components have a standard message format. Of course, all messages passing through the different layers of the architecture might not comply with the model, but rather the model provides the default business data interchange formats so that components need only to know (at most) their own message format and the default message format. The most common representation used for the canonical message model is as a set of XML schemas. This has the benefit of making the type and message definitions directly reusable in the *Web Service Definition Language* (WSDL) schemas that describe the exposed services. A canonical message model consists of the following:

- Defined set of types, elements, and attributes representing the business entities and their business attributes used in all messages. Each definition includes data types, formats, structures, names, and rules governing the allowable values of the type.
- Defined set of messages, each including a related set of the previously defined types, elements, and attributes structured to provide a business document with a specific semantic meaning and context.

A canonical message model should be derived from pertinent information and data models. The canonical data model describes the business entities, attributes, and relationships in a normalized form structured to reflect their business uses. Entities will have connections to other entities based on relationships. Exposure of a service interface must address how these entity relationships are exposed such that sets of required likely reusable information are exposed within the context of the business consumers to avoid proliferation of unnecessary information.

The canonical message model should be designed to support flexibility and extensibility such that the evolving business requirements on the architecture can be easily accommodated. Industry standards and variation analysis can be used to optimize the chosen formats. Implementation through XML schemas provides both the strong data typing rules and flexible structures needed to meet this goal.

Each service exposed in the SOA solution should have input and output messages that are defined directly by the canonical message model or that have a clear and explicit mapping to the canonical message model. This ensures structural and semantic interoperability across all the components participating in the SOA ecosystem.

The canonical message model does not define technical metadata such as routing or security information; it only defines business information. It is common to include technical metadata in the messages passed between systems. Typically, this technical detail can be isolated to message headers so as not to corrupt the business information in the message. Such techniques are common for handling security credentials, transaction states, routing information, message and service versioning, and so on. These metadata can be defined as enterprise messaging standards, but should be kept separate from the business information structures and semantics expressed in the canonical message model.

A canonical message model represents an agreement between different parties in an enterprise or line of business to transform local and often differing implementations or data structures and data sets into a common data format that can be utilized when processing a service. The input format for a message may be in a canonical data format and transformed from an underlying local implementation as the service passes from implementation to implementation, from component to component, which will leverage potentially differing underlying data models. The message format will leverage an underlying canonical message model that is accepted across the transaction path. This is akin to having a business process that cuts across multiple lines of business; as the process invokes services on each line of business, there is a common understanding of the overarching message model.

36. How Should a Canonical Message Model Be Created?

Several objectives for canonical message models influence how they should be created. That is, creating a canonical message model fulfills one or more of these objectives:

- Aligns the information exposed in service messages with defined information models where each message or message element is clearly defined for both structure and semantic meaning within the business context for which it is intended to be used
- Aligns the information exposed in service messages with the accepted business view developed in the logical data model, increasing reuse among service providers and consumers
- Accelerates the development of new messages by providing a standard set of information shared by all messages
- Increases efficiency of integration efforts by providing the default syntax and semantics for information exchange
- Reduces the complexity and frequency that data mapping rules are required to allow different SOA services and components to efficiently communicate
- Accelerates the definition and design of services by providing a set of reusable message constructs from which service interfaces can be composed

The starting point for the canonical message model is defining the data types and complex data types, which comprise the building blocks for messages. Data types can be derived directly from the logical data models. Attributes will map to either XML elements or XML attributes. Entities will map to XML elements. Rules such as value constraints, semantic metadata, and cardinality will also be propagated into the XML schema. The XML schema language cannot completely replicate the structure of a data model in terms of type hierarchies or cardinalities. Each XML message definition is restricted to a tree structure. At the same time, it may be futile to build a single XML message that traverses all the relationships and subtypes of the logical data model, because such a message would be difficult to construct and have no practical usage.

The next step is to identify the candidate message formats. The canonical message model provides a reusable set of types and messages, and defining the messages will require a balance of competing concerns. The resulting message set must be general enough to be reusable. Ideally, each message will be used by more than one service; however, each service must be able to construct messages from the message set that are appropriate for that service's interface. The message format must consider the range of potential message uses in the system along with the most likely areas of extensibility for the messages. Existing application programming interfaces, information models, data models, and the service model are all sources for formulating the candidate message formats.

The final step is to finalize the canonical messages. Using XML or an equivalent is a best practice for the implementation of the message design. Some organizations have created their own version of XML while retaining many of its core properties related to extensibility and flexibility. Using XML, practitioners will make design choices such as how to handle many-to-many and recursive relationships in existing models and other design choices.

Whenever data is shared, either horizontally or vertically, there must be a common understanding between the two participants of both the structure and the meaning of the data being exchanged. If a common data representation is agreed between the participants, such as passing XML messages as defined in the canonical message model, this task is trivial. However, in many cases, conformity to the canonical message model is not possible. For example, legacy applications, legacy databases, packaged applications, and external service providers will all have developed components without knowledge of the canonical message model. Hence, some aspects of the canonical message set may result from data mapping, where mapping occurs from each data format to the canonical message model. During runtime, each participant understands one external data format, which is the canonical message model.

37. Can SOA Improve Data Quality?

SOA is not a silver bullet for improving data quality. However, by understanding the operational data exposed by services, SOA

provides an opportunity to leverage services to improve data quality. The level of data quality required to effectively support business operations will vary by applications or lines of business, depending on the data needed to conduct that business unit's operations. For example, financial systems require a high degree of quality data (because of the importance and usage of the data), but a human resources system might have the latitude to operate with a lower level of data quality without significantly impacting business operations.

If data quality issues are present, organizations can pursue preventive data quality approaches that focus on the development of new data sources and integration services or pursue detective data quality approaches that focus on identification and remediation of poor data quality. Data quality detection, correction, prevention, and ongoing monitoring are beyond the scope of most SOA projects, but many of the architecture/application principles related to services can be applied to address data quality.

38. *What Are the Common Pitfalls with Information Architecture and SOA?*

The most common pitfall is not using information architecture to enhance aspects of SOA solutions. This includes failure to develop or use information models as a basis for message design. It includes not recognizing the need for a canonical message model for message design that is derived from information models. Information models should be leveraged when defining input message formats and output message formats. The service contracts will assume a certain message model that is passed to the service, whether on the input or the output. The *Web Service Definition Language* (WSDL) provides a definition of the service operations, the interface the input and output messages. Information models will facilitate the design of the input and output messages.

Combining information architecture and service-oriented architecture enables both to take advantage of the strengths of each. Information architecture benefits from the use of services, and SOA benefits from having a common understanding of business terms as expressed in information architecture models. Ignoring best practices of master data management, data cleansing, data transformations,

and data brokering or information architecture can lead to an infertile SOA adoption where information services are duplicated and no longer can be relied on for their quality.

Another common pitfall is not properly adopting information services as high-value business services versus services that simply retrieve or manipulate data (*create, read, update, and delete* [CRUD] services). Transformation of data from one format to another from one system to another is a key concern of information management. Services can externalize the transformation of formats; not only in the form of the notorious CRUD operations, which create, read, update, and delete information entities, but also to provide referential integrity, consistency, and replication of information and data as needed. Transformation of data from one format to another should be transparent to the consumer of information. Information services do just that—they create transparency for the consumer by shielding the consumer from the complexities of multiple heterogeneous systems and data sources. Information services are responsible for understanding the format in which the consumer needs the information and the consumer does not have to figure out what transformations should be applied to data to obtain clean or quality information.

Improving the effectiveness and efficiency of integration is a goal of many SOA adoptions and one that also requires the use of information architecture in areas of transformation. This represents another common pitfall when organizations don't use aspects of information architecture in this design aspect of the *enterprise service bus* (ESB). The process of transformation often requires an information mediator to access a set of rules and policies pertaining to the access rights, authorizations, and data format needs of the requesting process or service. An information broker or data broker will provide the mediation between the information requester and the information provider as a result of a service invocation.

Information: Key Concepts

Information architecture can leverage and utilize the best practices offered by service orientation to encapsulate information as a

service in such a way as to make access to information more loosely coupled, less platform dependent, less implementation dependent, and more consistently available across the organization for wherever the information is to be used.

Creating a canonical message model avoids rework and inconsistent message types, formats, and semantics in message exchanges and integration scenarios. Such models help to avoid the scenarios where each message will be defined strictly within the context and requirements of the message provider and consumer. Such a scenario can lead to proliferation of messages in the solution that offer little potential for reuse and carry a high maintenance cost.

Inconsistent message types, formats, and semantics in systems require that for each new integration scenario there will be a need to analyze the participants and develop message maps to address inconsistencies. The canonical message model plays a role in developing standard, reusable messages.

Brokering data or using services to provide mediation, transformation, and accessibility information to underlying data independent of location to authorized and authenticated users, whether a user interface or an application-to-application scenario, is a cornerstone of information as a service.

Information as a service is more than just a gateway function to underlying data sets. In addition to the transformation and rule application capabilities of an information service, information in the context of SOA may draw upon multiple and disparate and possibly geographically separated data sources. The ability to consolidate information from multiple sources or take a stream of incoming data and break it apart and assign it to multiple target data sources is also a factor of the intersection between SOA and information management.

INDEX

A

abstraction, levels of, 9
 aligned state (business and IT relationships), 22
 analytical services, 73
 APIs (application programming interfaces), 73
 service interfaces versus, 75
 architectural style, SOA as, 7–8
 asynchronous, defined, 11

B

barriers to SOA adoption,
 removing, 32–33
 bootstrapping, 59
 business processes
 defined, 11
 business services, information
 services as, 76. *See also* SOA
 services
 business stakeholders
 alignment with IT stakeholders, 20
 IT processes, relationship with, 24
 IT stakeholders, relationship
 with, 22

C

canonical message models
 creating, 82–83
 explained, 80–81

Centers of Excellence (CoEs),
 58–60
 centralized service delivery
 models (organizational
 structure), 26
 change management
 common pitfalls, 41–42
 organizational change
 management, role of, 30–32
 role of, 53–55
 classifying services, 39–40
 coarse-grained, defined, 11
 COBIT (Control Objectives for
 Information and related
 Technology), 47
 CoEs (Centers of Excellence),
 58–60
 communication processes, 61
 compliance processes, 61
 components, defined, 12
 composable, defined, 10
 composite services, 12
 constructs of SOA, 9–14
 consumers, defined, 11
 content services, 72
 Control Objectives for
 Information and related
 Technology (COBIT), 47
 corporate governance, 48
 CRUD (create, read, update,
 delete) operations by
 information services, 77

D

data models, information models
versus, 79

data quality in SOA, 83–84

data services, 72

DBA (database administration), 30

design-time governance, 56
versus runtime governance,
65–66

discoverable, defined, 10

E

EA (enterprise activity) versus
SOA governance, 61–63

enterprise activity (EA) versus
SOA governance, 61–63

Enterprise Architecture
Framework, 63

enterprise information models,
78–80

enterprise service bus
(ESB), 28, 31

ESB (enterprise service bus),
28–31, 54

exception and appeals
processes, 61

F

factory models (organizational
structure), 26

“fiefdom syndrome,” 19

funding for shared services,
33–36

G–H

governance
business and IT processes, 23
change management, role of,
53–55

corporate governance, 48
defined, 10
design-time governance, 56
IT governance, 46
overview, 46–48, 67–68
runtime governance, 56
SOA. *See* SOA governance, 46

I–J

identification of information
services, 76–77

information, as service, 69

information architects, defined, 69

information architecture
canonical message models
creating, 82–83
explained, 80–81

enterprise information models,
78–80

information services
CRUD operations by, 77
explained, 71–76
identification of, 76–77

pitfalls with SOA, 84–85

SOA and, 70–71

information flows, defined, 12

information models, data models
versus, 79

information services
classifying, 72
CRUD operations by, 77
explained, 71–76
identification of, 76–77

Information Technology
Infrastructure Library (ITIL), 47

integration centers
(organizational structure), 28–30

integration services, 72

integration technology. *See* EAI
(enterprise application
integration)

investment management,
business and IT processes, 23

IT (information technology)

governance, 46
in organizational structure,
24–30
processes, relationship with
business stakeholders, 24
stakeholders
*alignment with business
stakeholders, 20*
*relationship with business
stakeholders, 22*

ITIL (Information Technology
Infrastructure Library), 47

K–L

location transparent,
defined, 10
loose coupling
defined, 10

M

master data services, 73
measuring effectiveness
of SOA governance, 64–65
messages. *See* canonical message
models

N–O

objects, 12
organizational change
management
common pitfalls, 41–42
role of, 30–32
organizations
alignment between business and
IT stakeholders, 20
business and IT processes,
relationship between, 24

business and IT stakeholders,
relationship between, 22
common SOA adoption pitfalls,
41–42
funding for shared services,
33–36
getting started with SOA
governance, 46–53
prioritization for shared services,
37
removing barriers to SOA
adoption, 32–33
structure of, 24–30
owners of services, explained, 38
ownership of service reuse, 40–41

P

platform independent, defined,
10
portfolio management, business
and IT processes, 23
prioritization
for shared services, 37
processes. *See* business processes
project prioritization, business
and IT processes, 24
protocol independent, defined, 10

Q–R

quality of data in SOA, 83–84
queries by information services,
77
removing barriers to SOA
adoption, 32–33
requirements
for SOA implementations, 15–16
requirements gathering, business
and IT processes, 24
resistance to SOA adoption,
overcoming, 32–33

reuse
 ownership of, 40–41
 role of change management, 53–55
 Ross, Jeanne W., 47
 runtime governance, 56
 versus design-time governance,
 65–66

S

self-describing, defined, 10
 service centers, centralizing
 service development, 57
 service classification, 39–40
 Service Component Layer
 (architecture), 74
 service components, defined, 74–76
 service development, centralizing
 in service centers, 57
 service identification. *See*
 identification
 service interfaces
 APIs versus, 75
 service orientation, defined, 6.
See also SOA (service-oriented
 architecture)
 service owners, explained, 38
 service reuse, ownership of, 40–41
 service-oriented architecture. *See*
 SOA (service-oriented
 architecture)
 services, 9
 attributes of, 9–11
 composite services, 12
 defined, 11, 14, 74
 information services, 69
classifying, 72
CRUD operations by, 77
explained, 71–76
identification of, 76–77
 requirements for SOA
 implementations, 15–16

shared services
funding, 33–36
prioritization, 37
 SOA services, defined, 15
 web services, SOA services
 versus, 14

Services Layer (architecture), 74

shared service development
 models (organizational
 structure), 26–29

shared services

funding, 33–36
prioritization, 37

skunk work projects, 36

SOA (service-oriented
 architecture)

as architectural style, 7–8

CoEs (Centers of Excellence),
 58–60

constructs of, 9–14

data quality and, 83–84

explained, 5–8

funding models, 67

governance

*CoEs (Centers of
 Excellence), 58–60*

EA (enterprise activity)

versus, 61–63

getting started, 46–53

*implementation of SOA tools
 and, 55–56*

*measuring effectiveness of,
 64–65*

overview, 46–53

pitfalls of, 66–67

service development,

centralizing in service, 57

*success of SOA projects,
 63–64*

implementations, requirements
 for, 15–16

information architecture and,
70–71

pitfalls

- with information architecture, 84–85*

projects

- success of, 63–64*

removing barriers to, 32–33

services

- defined, 15*
- requirements for, 15*
- web services versus, 14*

stakeholders in, 6–7

stakeholders

- in SOA, 6–7

stateless, defined, 9

strategic planning, business and IT processes, 23

structure of organizations, 24–30

success of SOA projects (SOA governance), 63–64

synced state (business and IT relationships), 21

T

technologies in SOA, 13–14

tools, implementing SOA tools (governance), 55–56

U–V

unified state (business and IT relationships), 21–23

vitality processes, 61

W–Z

web services

- requirements for, 15
- SOA services versus, 14

Weill, Peter, 47

Content Excerpted from
Pearson Technology Group by IBM.
Not for Resale or Distribution.