Open Group Standard

SOA Reference Architecture

Technical Standard

SOA Reference Architecture

ISBN: 1-937218-01-0

Document Number: C119

Published by The Open Group, November 2011.

Comments relating to the material contained in this document may be submitted to:

The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, United Kingdom

or by electronic mail to:

ogspecs@opengroup.org

Contents

List of Figures

List of Tables

Preface

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 400 member organizations, The Open Group has a diverse membership that spans all sectors of the IT community – customers, systems and solutions suppliers, tool vendors, integrators, and consultants, as well as academics and researchers – to:

- Capture, understand, and address current and emerging requirements, and establish policies and share best practices
- Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
- Offer a comprehensive set of services to enhance the operational efficiency of consortia
- Operate the industry's premier certification service

Further information on The Open Group is available at www.opengroup.org.

The Open Group publishes a wide range of technical documentation, most of which is focused on development of Open Group Standards and Guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/bookstore.

Readers should note that updates – in the form of Corrigenda – may apply to any publication. This information is published at www.opengroup.org/corrigenda.

This Document

This document is The Open Group Standard for SOA Reference Architecture (SOA RA). It has been developed by the SOA Reference Architecture project of the SOA Work Group within The Open Group.

Trademarks

Acknowledgements

The Open Group gratefully acknowledges the contribution of the following people in the development of this document:

- Ali Arsanjani, IBM
- Tony Carrato, IBM
- Jorge Diaz, IBM
- Awel Dico, Bank of Montreal
- Michael Ellis, Independent Consultant
- Mats Gejnevall, Capgemini
- Shuvanker Ghosh, IBM
- Chris Harding, The Open Group
- Heather Kreger, IBM
- Nikhil Kumar, ApTSi
- Joost van der Vlies, HP
- Liang-Jie Zhang, Kingdee

From IBM, the project team would like to acknowledge input and advice from the following colleagues: Raj Cherchatil, Arnauld Desprets, Jorge Diaz, Marc Fiammante, Donald Ferguson, Greg Flurry, Rolando Franco, Biffle French, George Galambos, Joe Hardman, Andrew Hately, Rob High Jr., Kerrie Holley, David Janson, Petra Kopp, Rao Kormili, Min Luo, Stefan Pappe, Emily Plachy, Siddarth Purohit, Robert Rafuse, Rachel Reinitz, Rick Robinson, Tony Storey, Raghu Varadan, Jamshid Vayghan, Bobby Wolf, Dan Wolfson, Olaf Zimmerman, Jia Zhang, and other members of the IBM worldwide SOA community of practice.

From ApTSi, the project team would like to acknowledge input and advice from Dinakar Sosale, Thomas Osberg, Anne Bonnar, Neb Brankovic, and John Mathew.

Referenced Documents

The following documents are referenced in this Technical Standard:

[1] Ali Arsanjani: Service-Oriented Modeling and Architecture: How to Identify, Specify, and Realize your Services, IBM developerWorks; refer to: www-128.ibm.com/developerworks/webservices/library/ws-soa-design1.

[2] Ali Arsanjani, Liang-Jie Zhang, Abdul Allam, Michael Ellis, et al: Design an SOA Solution using a Reference Architecture, IBM developerWorks; refer to: www.ibm.com/developerworks/library/ar-archtemp.

[3] Ali Arsanjani, Shuvanker Ghosh, et al: SOMA: A Method for Developing Service-Oriented Solutions, IBM Systems Journal, Volume 47, No. 3, 2008, p.377-396.

[4] Liang-Jie Zhang and Bing Li: Requirements-Driven Dynamic Services Composition for Web Services and Grid Solutions, Journal of Grid Computing 2(2): 121-140 (2004).

[5] Donald Ferguson and Marcia Stockton: SOA Programming Model; refer to: www-128.ibm.com/developerworks/webservices/library/ws-soa-progmodel/.

[6] Web Services Description Language (WSDL), Version 2.0, Part 1: Core Language, Editors: R. Chinnici, J-J. Moreau, A. Ryman, S. Weerawarana, World Wide Web Consortium, 27 March 2006. This version of the specification is available at www.w3.org/TR/2006/CR-wsdl20-20060327. The latest version is available at www.w3.org/TR/wsdl20.

[7] M. Shaw and D. Garlan: Software Architecture: Perspectives on an Emerging Discipline, Prentice-Hall, 1996.

[8] Ali Arsanjani: Explicit Representation of Service Semantics: Towards Automated Composition Through a Dynamically Re-Configurable Architectural Style for On-Demand Computing (p.34-37), ICWS, 2003.

[9] Gordon Bell, Allen Newell, and Daniel Sieworek: Computer Structures: Principles and Examples (Chapter 3), McGrawHill, 1982.

[10] TOGAF Version 9, Enterprise Edition; refer to: www.opengroup.org/togaf.

[11] SOA Definition, The Open Group SOA Work Group, June 2006; refer to: www.opengroup.org/projects/soa.

[12] Web Services Policy (WS-Policy), Version 1.5, W3C Recommendation, Editors: A. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P.Yendluri, T. Boubez, U. Yalcinalp, World Wide Web Consortium, 04 September 2007; refer to: www.w3.org/TR/2007/REC-ws-policy-20070904/.

[13] OASIS Open Composite Services Architecture (Open CSA), Member Section, OASIS; refer to: www.oasis-opencsa.org.

[14] Web Services for Remote Portlets (WSRP), Version 1.0, OASIS, August 2003; refer to: www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf.

[15] Web Services Business Process Execution Language, Version 2.0 (WS4BPEL), OASIS Standard, April 2007; refer to: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf.

[16] Voice Extensible Markup Language (VoiceXML), W3C Recommendation, March 2004; refer to: www.w3.org/TR/voicexml20.

[17] Java 2 Platform Enterprise Edition (J2EE), Sun Microsystems, San Jose, CA; refer to: http://java.sun.com/javaee.

[18] .NET Framework, Microsoft, Redmond, WA; refer to: www.microsoft.com/NET.

[19] XML Transformations (XSLT), Version 1.0, W3C Recommendation, November 1999; refer to: www.w3.org/TR/xslt.

[20] J.J. Garrett: Ajax: A New Approach to Web Applications, February 2005, retrieved on June 19, 2008 from: Adaptive Path.com.

[21] The Open Group Service Integration Maturity Model (OSIMM), Technical Standard, October 2010 (C092); refer to: www.opengroup.org/bookstore/catalog/c092.htm.

[22] Navigating the SOA Open Standards Landscape Around Architecture, Joint White Paper from OASIS, OMG, and The Open Group, July 2009 (W096); refer to: www.opengroup.org/bookstore/catalog/w096.htm.

[23] OASIS Reference Model for SOA (SOA RM), Version 1.0, OASIS Standard, 12 October 2006; refer to: docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

[24] The Open Group SOA Ontology, Technical Standard, October 2010 (C104); refer to: www.opengroup.org/bookstore/catalog/c104.htm.

[25] The Open Group SOA Governance Framework, Technical Standard, August 2009 (C093); refer to: www.opengroup.org/bookstore/catalog/c093.htm.

[26] Using TOGAF® to Define and Govern SOAs; The Open Group Guide, May 2011 (G113); refer to: www.opengroup.org/bookstore/catalog/g113.htm.

Informative References

- Ali Arsanjani, Liang-Jie Zhang, Abdul Allam, Michael Ellis, et al: S3: A Service-Oriented Reference Architecture, IT Professional, Volume 9, Issue 3, May-June 2007, p.10-17.

- D.L. Parnas, P.C. Clements, D.M. Weiss: The Modular Structure of Complex Systems, IEEE Transactions on Software Engineering, SE-11(3), 1985, p.259-266.
- Liang-Jie Zhang, Jia Zhang: Design of Service Component Layer in SOA Reference Architecture, COMPSAC (1) 2009: 474-479.
- Liang-Jie Zhang, Jia Zhang: Componentization of Business Process Layer in the SOA Reference Architecture, IEEE SCC 2009: 316-323.
- Liang-Jie Zhang, Jia Zhang: An Integrated Service Model Approach for Enabling SOA, IT Professional 11(5): 28-33 (2009).
- Liang-Jie Zhang, Jia Zhang, Abdul Allam: A Method and Case Study of Designing Presentation Module in an SOA-based Solution Using Configurable Architectural Building Blocks (ABBs), IEEE SCC (2) 2008: 459-467.

1 Introduction

1.1 Objective

A Service-Oriented Architecture (SOA) facilitates the creation of flexible, re-usable assets for enabling end-to-end business solutions. Increasingly, companies are adopting the principles and techniques associated with SOA for different types of projects in different industries worldwide.

The usage of the SOA Reference Architecture (SOA RA) is a key enabler for the achievement of the value propositions of an SOA.

This specification presents an SOA RA, which provides guidelines and options for making architectural, design, and implementation decisions in the implementation of solutions. The goal of this SOA RA is to provide a blueprint for creating or evaluating architecture.

Additionally, it provides insights, patterns, and the building blocks for integrating fundamental elements of an SOA into a solution or enterprise architecture.

Informally, the aim of the SOA RA is to answer some of the key questions and issues encountered by architects, including but not restricted to common questions such as:

- What are the aspects, building blocks, and layers of an SOA that I need to consider in designing solutions, establishing enterprise architecture guidelines, or assessing an architecture based on service-oriented principles?
- What are the building blocks I need to include in each layer of my solution or standardize as part of a enterprise architecture?
- What are some of the key architectural decisions I need to make when designing a solution, or assessing an architecture that is based on service-oriented principles?
- How do I increase my chances of gaining benefit from using SOA by taking into account key layers and building blocks with which I may initially be unfamiliar as our company makes it journey through higher levels of maturity?[1]
- Which roles in a project would benefit from using these principles and guidelines?

The SOA RA is used as a blueprint and includes templates and guidelines for enterprise and solution architects as well as software engineering roles within the software development life-cycle. These facilitate and ultimately enable automation and streamlining the process of modeling and documenting the architectural layers, the capabilities and the Architecture Building Blocks (ABB) within them, options for layers and ABBs, mapping of products to the ABBs, and architectural and design decisions that contribute to the creation of an SOA. It is

[1] One of the ways in which we can establish a baseline and move to higher levels of maturity is to use The Open Group Service Integration Maturity Model (OSIMM) [21].

intended to support organizations adopting SOA, product vendors building SOA infrastructure components, integrators engaged in the building of SOA solutions, and standards bodies engaged in the specifications for SOA.

1.2 Overview

In this Open Group standard we present the results of abstracting and using an SOA RA based on multiple projects in different industries across the world.

During these projects, a number of key underlying themes have emerged that have been formalized into a meta-model for the SOA RA, as shown in Figure 3.

This meta-model defines a number of layers, ABBs, architectural and design decisions, patterns, options, and the separation of concerns needed to design or evaluate architecture.

Furthermore, these elements represent the results of applying an SOA method to identify, specify, realize, and implement services, components, and flows of an end-to-end solution in the context of a service-oriented approach. The SOA RA provides common systems architecture [7], and the mechanism to translate it to industry architecture and individual solution architecture, as well as providing traceability between these specializations of architecture.

Thus, the SOA RA also acts as a communication vehicle for organizations to use to provide a high-level specification of what SOA components are and how to pick specific solutions, and a mechanism to align technology with business requirements.

The SOA RA, being composed of a collection of layers, uses an approach in which each layer provides a set of "capabilities" that are realized by a set of ABBs. The ABB or group of ABBs may be implemented in a target implementation platform or product by multiple vendors.

This approach allows users of the SOA RA to look for a set of capabilities and assess or create an architecture that realizes those capabilities using a set of logical building blocks without tying those building blocks to a specific vendor implementation.

It thus allows the user of the standard to align the technical capabilities with business capabilities and requirements, without necessarily implying implementation decisions. The SOA RA then provides the ABBs that will realize those capabilities, thus acting as a mechanism for initially deferring and then explicitly making implementation decisions to indicate how a given component of the solution, the ABB, will choose to be implemented on their project.

The SOA RA is based on establishing the building blocks of SOA: services, components, and flows (processes) that collectively support business processes and business goals. The metadata underlying each layer and relationship between layers can further facilitate in bridging the gap between business and IT from solution modeling to solution realization.

Another major capability afforded by the SOA RA is the increase in re-usability when designing and developing solution assets for rapid development, deployment, and management of SOA solutions within industry or across industries.

This document is organized as follows:

- Chapter 1 provides a general introduction.
- Chapter 2 provides motivation for using the SOA RA.
- Chapter 3 provides an overview of the principles on which the SOA RA was derived.
- Chapter 4 overviews the basic concepts of the SOA RA.
- Chapter 5 reviews the layers of the SOA RA in detail.
- Chapter 6 explains capabilities in the SOA RA.
- Chapter 7 describes assumptions for the layers of the SOA RA.
- Chapters 8 through 16 each explain one of the nine layers of the SOA RA.
- Chapter 17 outlines related work and uses of the SOA RA.
- Appendix A relates the SOA RA to other SOA standards.
- Appendix B relates the SOA RA to The Open Group Guide: Using TOGAF to Define and Govern SOAs.

1.3 Conformance

Intentionally left blank in this version.

1.4 Terminology

Can Describes a permissible optional feature or behavior available to the user or application. The feature or behavior is mandatory for an implementation that conforms to this document. An application can rely on the existence of the feature or behavior.

Implementation-dependent
(Same meaning as "implementation-defined".) Describes a value or behavior that is not defined by this document but is selected by an implementer. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementer shall document such a value or behavior so that it can be used correctly by an application.

Legacy Describes a feature or behavior that is being retained for compatibility with older applications, but which has limitations which make it inappropriate for developing portable applications. New applications should use alternative means of obtaining equivalent functionality.

May — Describes a feature or behavior that is optional for an implementation that conforms to this document. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations. To avoid ambiguity, the opposite of "may" is expressed as "need not", instead of "may not".

Must — Describes a feature or behavior that is mandatory for an application or user. An implementation that conforms to this document shall support this feature or behavior.

Shall — Describes a feature or behavior that is mandatory for an implementation that conforms to this document. An application can rely on the existence of the feature or behavior.

Should — For an implementation that conforms to this document, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations. For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

Undefined — Describes the nature of a value or behavior not defined by this document that results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

Unspecified — Describes the nature of a value or behavior not specified by this document that results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

Will — Same meaning as "shall"; "shall" is the preferred term.

1.5 Future Directions

Subsequent drafts of the SOA RA may include examples of application of the standard. There is no commitment to any particular additional content and other content not mentioned here may be added.

2 Motivation

Over the past 25 years, software architecture has grown rapidly as a discipline. With the pioneering work of Bell, Newell, and Sieworek [9] (Processor Memory Switch (PMS) notation and the birth of Architecture Description Languages (ADLs)), Shaw and Garlan on Software Architecture [7], and later additions, we recognize the need for architectural patterns and styles. An architectural style is the combination of distinctive features in which architecture is performed or expressed [8]. This can be visualized as a set of components (we will call Architecture Building Blocks (ABBs) to distinguish the generic nature of these), connectors, constraints [5], composition, containers, and configurations. Arsanjani [1] refers to these as the 6 Cs of an architectural style [6]. We know that a software system often has a predominant architectural style for its design. It has been shown that modularity or decoupling facilitates the creation of complex systems [4], such as that required by today's business applications.

In recent years, the decoupling of interface from implementation at the programming level has been elevated to an architectural level by loosely coupling based on open standards the interface of a service consumed by a service consumer from its implementation by a service provider and by decoupling the implementation from its binding. This concept is reflected in a layer in the architecture corresponding to each of these notions; i.e., a layer for services and a layer for the runtime operational systems.

This style of architecture has come to be known as Service-Oriented Architecture (SOA), where rather than the implementations of components being exposed and known, only the service interfaces provided are published and consumers are insulated from the details of the underlying implementation by a provider.

Thus, an SOA enables business and IT convergence through agreement on a (contract consisting of a) set of business-aligned IT services that collectively support an organization's business processes and business goals. Not only does it provide flexible decoupled functionality that can be re-used because it is based on open standards, but it also provides the mechanisms to externalize variations [6] of Quality of Service (QoS) in declarative specifications such as WS-Policy [12] and related standards.

2.1 Key Business Benefits of SOA

As a flexible and extensible architectural framework, SOA has the following defining capabilities:

- *Reducing Cost:* Through providing the opportunity to consolidate redundant application functionality and decouple functionality from obsolete and increasingly costly applications while leveraging existing investments.

- *Agility:* Structure business solutions based on a set of business and IT services in such as way as to facilitate the rapid restructuring and reconfiguration of the business processes and solutions that consume them.
- *Increasing Competitive Advantage:* Provide the opportunity to enter into new markets and leverage existing business capabilities in new and innovative ways using a set of loosely-coupled IT services. Potentially increase market share and business value by offering new and better business services.
- *Time-to-Market:* Deliver business-aligned solutions faster by allowing the business to decide on the key drivers of a solution and allowing IT to rapidly support and implement that direction.
- *Consolidation:* Integrate across silo'ed solutions and organizations, reduce the physical number of systems, and enable consolidation of platforms under a program of "graceful transition" from legacy spaghetti dependencies to a more organized and integrated set of coexisting systems.
- *Alignment:* SOA enables organizations to better align IT to business goals, enabling the business to associate IT with capabilities that an organization wants to achieve in alignment with its strategic plan, leading to both sustained agility and re-use over time.

However, significant challenges in creating an SOA solution still remain: correct service identification, selection, design, solution element selection and combination, modeling of services, governance, interoperability, and the ability to identify different components key to the effective design, usage, and evolution of SOA. For example, from a technical perspective, the architect needs to answer questions such as:

- What are the considerations and criteria for producing an SOA solution?
- How can an SOA solution be organized as an architectural framework with inter-connected architectures and transformation capabilities?
- How can an SOA solution be designed in a manner that maximizes asset re-use?
- How can automated tools take the guesswork out of architecture validation and capacity planning?

In order to address these issues, this standard presents a reference architecture for SOA-based solutions. It provides a high-level abstraction of an SOA partitioned and factored into layers, each of which provides a set of capabilities required to enable the working of an SOA. Each layer addresses a specific subset of characteristics and responsibilities that relate to unique value propositions within an SOA. As stated above, underlying this layered architecture is a meta-model consisting of layers, capabilities, ABBs, interactions, patterns, options, and architectural decisions and the relation between capabilities, ABBs, and layers. These will guide the architect in the creation and evaluation of the architecture. Note that per TOGAF, Chapter 3 (Definitions):

"*(Capabilities) … are an ability that an organization, person, or system possesses (to deliver a product or service)*" [10]

Likewise, an ABB represents a basic element of re-usable functionality, providing support for one or more capabilities, that can be realized by one or more components or products; examples of the responsibilities of an ABB include: service definition, mediation, routing, etc.

3 Key Principles

The SOA Reference Architecture (SOA RA) has been defined and refined with consideration for the following principles:

1. The SOA RA should be a generic solution that is vendor-neutral.
2. The SOA RA is based on a model of standards compliance.
3. The SOA RA should be capable of being instantiated to produce:
    a. Intermediary industry architectures
    b. Concrete solution architectures
4. The SOA RA should promote and facilitate IT-to-business alignment.
5. The SOA RA should address multiple stakeholder perspectives.
    c. For organizations implementing the SOA RA within their enterprise:
        — The SOA RA should be generic enough to be independent of vendor solutions.
        — The SOA RA should define standard capabilities, building blocks, architectural decisions, and other attributes to create a framework of understanding sufficient to enable an assessment of conformance.
    d. For product vendors:
        — The SOA RA should provide a set of standards and enough specificity that they can use it to drive evaluation of compliance with those underlying standards.
    e. For integrators:
        — The integrator should be able to use it as a model to define specific constraints and directions for SOA implementations.
    f. For standards bodies:
        — The SOA RA should provide a reference against which standards can be extended, or guidelines provided, more detailed levels of specificity can be defined, etc.
6. The manner in which the SOA RA is instantiated should be determined by the user of the SOA RA.
7. The SOA RA should use the fewest number of layers to depict the possible combinations and elements of an SOA.

4 Basic Concepts

A number of key and fundamental concepts recur throughout the SOA Reference Architecture (SOA RA). We will summarize them here.

4.1 Logical *versus* Physical Architecture

The distinction between logical/design-time and physical/runtime elements of the SOA are described below:

- All runtime elements that are part of the physical or operational or deployment architecture are actually running in the Operational Systems Layer of the SOA RA.
- We will provide a logical abstraction of the runtime at a moment in time and expand that view in terms of a set of layers that are depicted by horizontal/functional layers and a set of supporting or cross-cutting layers (backdrop) of the SOA RA. The five horizontal layers deal with enabling the business capabilities required by the application running on the architecture. The four supporting layers support the functionality that is provided by the horizontal layers. The horizontal or functional layers include the Service Component Layer, the Services Layer, the Business Process Layer, and the Consumer Layer.
- Each aspect of the runtime is abstracted into a layer whose responsibility is significantly distinct from that of the other layers. For example, the Consumer Layer is responsible for interaction with consumers of the service, whereas the Service Component Layer is responsible for the implementation of the service in a Service Component. That Service Component in turn will be running in a container within the Operational Systems Layer, at runtime.

4.2 Foundational Concepts

This section explains some of the concepts, principles, and reasoning of the SOA RA and sets the tone for the rest of the document. We recommend you read this before you read the SOA RA details, but it can be referred to at any time. Some of the terms in this section are defined later in the document. These foundational concepts will be discussed in a Question and Answer (Q&A) format.

Question 1

Does the Consumer Layer consist primarily of things like a Graphical User Interface (GUI), or is it a programmatic means of access to the services, or both?

It is both. The Consumer Layer provides access to services. It allows consumers to consume services. The means of consumption of services can be a GUI or an API-like connection to the services.

Question 2

Can the Integration Layer access services?

Yes. In fact, the Integration Layer is a layer of choice through which services are invoked, but it is not the only means to invoke services. Services can, in a less mature SOA, be invoked directly by the consumer without the level of indirection associated with an Integration Layer.

Question 3

Which layer is responsible for invoking the service end-point?

The Consumer Layer is responsible for invoking the service end-point. It accesses the services via the Integration Layer. Or it can access the service directly if the given architecture does not wish to implement an Integration Layer. Invocation is done by opening up access to the Consumer Layer.

Question 4

Who provisions the service?

Provisioning means doing all of the tasks necessary to make the service available for consumers to be able to invoke. Part of the responsibility of provisioning is updating the registry (in the Governance Layer) which contains the service metadata that consumers need to find, bind, and invoke services.

Question 5

Are all the Architecture Building Blocks (ABBs) of a layer required for every SOA implementation? Or can we pick and choose from among the ABBs based on the needs of individual projects?

Not all ABBs are required for every single implementation of an SOA. Instead every project will choose from the list of building blocks within each layer of the SOA RA and select those that are appropriate for the particular project. In the case where SOA RA is applied for the enterprise architecture standard within an organization, there will be patterns of ABBs within the layers that will be selected. Some will be mandatory and some will be optional, and projects will be chosen to conform to a fixed set of patterns and configurations of the ABBs along with product selections and implementations.

Question 6

Registries and repositories may need to exist in multiple layers during a physical implementation in my projects. Where are they considered to be in the SOA RA?

We have chosen to organize the registry and repository location in the Governance Layer of the SOA RA. In this way, we can manage, monitor, and administer the registry and/or repository from a single logical location, even though physically we may have federation or distribution.

Question 7

Can you tell us about how the nine layers would be utilized in practice for realization of the architecture?

The four vertical or cross-cutting layers – namely Integration Layer, Quality of Service Layer, Information Layer, and Governance Layer – are basically supporting capabilities realized through implementation and vendor products. The five functional or horizontal layers – namely Operational Systems Layer, Service Component Layer, Services Layer, Business Process Layer, and Consumer Layer – will support the functional capabilities of the architecture. The Operational Systems Layer will provide the actual runtime for all layers, vertical or horizontal.

Question 8

What are the ABBs in each layer?

They are the key building blocks of a particular layer. You can think of them as the components providing key capabilities that are expected from that layer. For example, the Integration Layer is expected to provide meditation, routing, and protocol transformation capabilities. Therefore, these capabilities are realized by a set of building blocks, each of which provides exactly that atomic unit of architectural capability you are seeking.

Question 9

What kind of ABBs do we have?

We have some ABBs that are related to the functionality within an application such as:

- Consumer Layer: Portal to loan processing application.
- Business Process Layer: Initiation of a loan processing application.
- Services Layer: Services that are required to support the loan processing application.
- Component Layer: Software components that need to be built that support the realization and implementation of the services.
- Operational Systems Layer: Actual runtime environment in which the components, legacy systems, all back-end applications, and packaged applications reside and run.

Then we have the cross-cutting layers which include the Integration Layer that serves as the central point where integration occurs across the enterprise and consolidates the design decisions into a set of software components that facilitate and enable the actual integration of applications.

The Information Layer includes all data and information needed to support and instantiate each of the layers. Note that it is a cross-cutting layer indicating that each of the horizontal layers may

have and will have data associated with their functioning and they will draw upon this data from the Information Layer via metadata, actual data, or analytics.

The Quality of Service Layer ensures Quality of Service (QoS) by serving as a collection point for the administration and control, or monitoring and management of most if not all of the Non-Functional Requirements (NFRs). This includes security, availability, configuration, monitoring, and management capabilities, to name a few.

Lastly, the Governance Layer provides a central point in which policies are set into registries and repositories. In general, governance capabilities and processes are administered and run centrally via this layer. Note again that this layer is the underlying layer for all of the other layers within the architecture and it relates to and touches upon all other functional and cross-cutting layers of the SOA RA.

Question 10

Where are the policies defined and where are they enforced?

Policy definition is the responsibility of the Governance Layer. The definition of Policy Enforcement Points (PEPs) and policy enforcers is the responsibility of the individual horizontal/functional layers, such as Services Layer, Business Process Layer, etc. The enforcement of those policies is then the responsibility of a cross-cutting layer, and we have chosen to consolidate that within the Quality of Service Layer. Therefore, monitoring and enforcement of the policies are the responsibility of the Quality of Service Layer, while the administration of the policies remains the responsibility of the Governance Layer.

Question 11

Where are the business rules defined?

Business rules are also a cross-cutting architectural concern. They need to be consistently applied across the multiple layers in the SOA or, over time, there is a tendency to develop rule divergence and lose the consistency that SOA brings. Therefore, business rule definition and management is the responsibility of the Governance Layer and its validation and enforcement is the responsibility of the Quality of Service Layer.

Question 12

Are ABBs in the SOA RA and artifacts that we create as part of methods the same?

Not in all cases. For example, a process definition or a process model artifact is used by the Business Process ABB in the Business Process Layer to describe the underlying business process.

Question 13

Events are cross-cutting. Where do they reside and where are they handled?

We have agreed to treat the cross-cutting nature of events within the Integration Layer, Quality of Service Layer, and Information Layer.

Question 14

How and where are complex event processing capabilities addressed? Which layer is responsible for the events?

No single layer is responsible for events. Many layers and corresponding ABBs collaborate to produce a composite capability related to events such as Complex Event Processing (CEP), Business Activity Monitoring (BAM), etc.

Business events occur during the course of business process execution. An example of this is when we want to monitor fraud. Fraud management occurs when during the execution of a business process certain data items' sequences and patterns are spotted that trigger a set of rules relating to possible fraud. This will trigger a notification and subsequent action. A set of data access or update patterns can also be a reason to trigger events in the Information Layer.

Thus, CEP is addressed across multiple layers of the SOA RA; it may start from the Business Process Layer or Information Layer. But the building blocks for producing and listening for events reside within the Integration Layer. The Quality of Service Layer is responsible for monitoring and managing the events. The Information Layer is used to define the events.

Question 15

If someone asserts they have an SOA that conforms to the SOA RA, what would have to be true for that statement to be true? Is the SOA RA normative? How should the SOA RA be used?

The SOA RA is intended to be a set of best practices and guidance on creating successful architectures using the SOA paradigm. It is not intended to be mandatory or normative.

It is a qualitative standard in that users of the SOA RA may choose to deviate from the standard in certain areas. Based on the maturity of the organization implementing SOA across the multiple domains (business, governance, methods and processes, application, architecture, information, and infrastructure), an organization may choose among the various ABBs provided by the SOA RA for conducting assessments, designing, or implementing architectures.

Some of the ABBs in the SOA RA are foundational and may be essential or mandatory, and others are advanced or optional and may be required for higher maturity in SOA.

If we claim we have an SOA then we should compare it to the reference architecture. If the SOA solution does not conform to the partial layer architecture and if there are certain things that are logically missing, then we should identify the missing building blocks that are key for the specific instance of the architecture. The corresponding Solution Building Blocks should be present.

There are implications in terms of resource requirements. We will see in RFPs compliance with the SOA RA. There will be a spectrum of implementation ranging from point-to-point to physically brokered via one or more Enterprise Service Buses (ESBs). Customers' architectures can be different but conformant. It is also a quality and compliance checking mechanism.

Question 16

What is the difference between an ABB, Service Component, Deployment Unit, Solution Component, and a Solution Building Block?

An ABB is a logical entity. Each layer is composed of ABBs with which it implements its capabilities.

We can separate ABBs into those that provide the infrastructure to support the SOA, and the services themselves. An example of an ABB which provides infrastructure support could be a Service Container ABB in the Services Layer, or a Mediator ABB in the Integration Layer. An example of an ABB related to implementing or providing a service is the Service Component ABB.

Let us understand the ABBs which form a service first. Each service has a Service Component – which is an ABB in the Service Component Layer. A Service Component is composed of a Functional and a Technical Component. The Functional Component provides the functional capability that the service implements. This could be a legacy system invocation, a Java POJO (Plain Old Java Object), a COBOL subroutine, etc., or one or more other Functional Components. The Technical Component encapsulates the technical capabilities to support standards compliance and technical support that a service must provide. However, the Service Component and its associated Technical and Functional Components are logical, design-time assets.

Now let us look at the creation of a service and its conversion into a runtime entity. The service is created using an Integrated Development Environment (IDE) of some sort, with resulting Service, Technical, and Functional Components. Next the Service, Functional, and Technical Components are bundled up into a Deployment Unit. Finally the Deployment Unit is deployed into a runtime environment, becoming a Solution Component. Figure 1 shows the structural relationship between the different service-related ABBs.

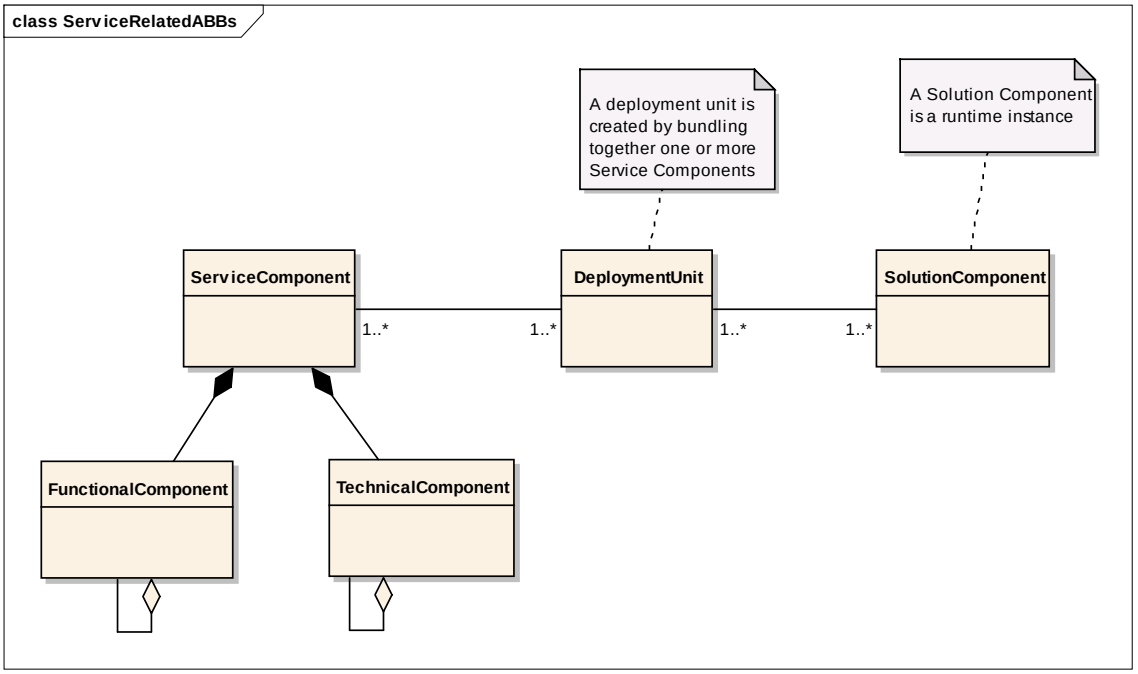


Figure 1: Structural Relationship between Service-Related ABBs

Figure 2 shows the "lifecycle", or the dynamic relationship between the design and development time Service, Functional, and Technical Components, the bundled Deployment Unit, and the runtime Solution Component.

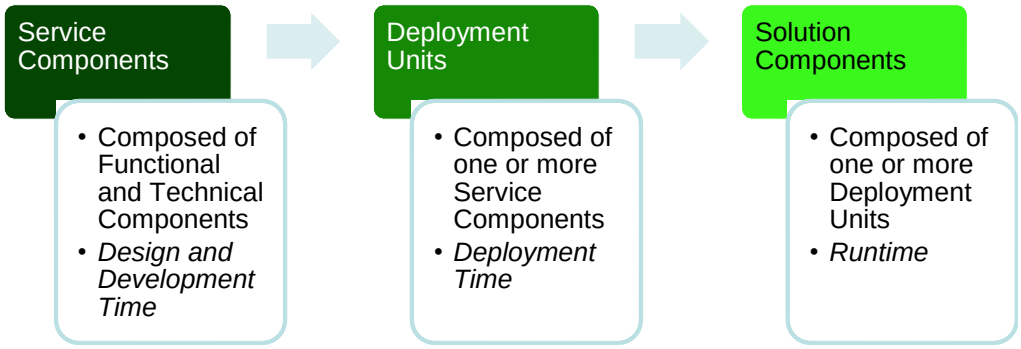


Figure 2: The Dynamic/Temporal Relationship between Service Components, Deployment Units, and Solution Components

To understand this in more detail, let us consider an example. In this example we have a CreditCheck Service, which is a SOAP doc-style Web Service, which checks credit by invoking other services – CheckInternalCredit, CheckCreditAcmeRatingAgency1, and CheckCreditAcmeRatingAgency2. CheckInternalCredit is a service which invokes a JEE component which encapsulates a mainframe-based relational database.

Let us consider CheckCredit first. We develop a Service Component which deploys on a .NET environment, whose Technical Component encapsulates the SOAP binding with the .NET stack and integration with the QoS implementation interfaces. The Functional Component for CheckCredit is a POCO (Plain Old CLR Object), which contains the logic to invoke the services CheckInternalCredit, CheckCreditAcmeRatingAgency1, and CheckCreditAcmeRatingAgency2, as well as to compose a response (the data) which is going to be sent back to the service consumer of CheckCredit. The build process compiles and bundles all the resources and parts of the Service Component together to form the .zip file and .dll component which is then placed in the runtime environment. When placed in the runtime environment (in the appropriate directory), the .dll component becomes the Solution Component for the CheckCredit service.

In practice, SOA typically involves multiple platforms and environments, so we will consider CheckInternalCredit next. This service is a SOAP doc-style web service using JAX-WS as the binding, a POJO (Plain Old Java Object) to wrap invocations to the database and build (compose) the response. In this case the Service Component includes the Technical Component which manages the SOAP stack binding and support, and the POJO, database components, and other elements which form the Functional Component. When the build script compiles and puts all the parts of the Service Component together it creates a Deployment Unit (let us call it a servlet for this discussion), which is then deployed into the appropriate directory to be used at runtime – becoming a Solution Component.

So what is a Solution Building Block? In both services considered in the example above, the service gets deployed into a Solution Building Block which provides the runtime implementation of a Service Container. In the case of the .NET service (CheckCredit), the Solution Building Block is (for the sake of our example) an IIS/WAS (Internet Information Services (IIS)/Windows Activation Service (WAS)), and in the case of the JEE service (CheckInternalCredit), the Solution Building Block is (for the sake of our example) an Acme JEE Server. In both cases, the corresponding ABB for the Solution Building Block is the Service Container ABB in the Services Layer.

As can be seen, the Solution Building Block is the runtime instance of an ABB that provides the infrastructure to run services. Solution Components on the other hand are ABBs which provide the runtime instances of the services themselves.

5 Overview of the SOA RA Layers

The SOA Reference Architecture (SOA RA) has nine layers representing nine key clusters of considerations and responsibilities that typically emerge in the process of designing an SOA solution or defining an enterprise architecture standard. Also, each layer is designed to correspond to reinforce and facilitate the realization of each of the various perspectives of SOA business value discussed in Section 2.1.

Taking a pragmatic approach, for each layer, we postulate three aspects that should be supported by the SOA RA: requirements (exemplified by the capabilities for each layer), logical (exemplified by the Architecture Building Blocks (ABBs)), and physical (this aspect will be left to the implementation of the standard by an adaptor of the standard using Solution Building Blocks).

The requirements aspect reflects what the layer enables and includes all of its capabilities; the logical aspect includes all the ABBs, design decisions, options, Key Performance Indicators (KPIs), etc.; while the physical aspect of each layer includes the realization of each logical aspect using technology, standards, and products that are determined by taking into consideration the different architectural decisions that are necessary to be made to realize and construct the architecture. The actual realization by a set of products or platform (i.e., Solution Building Blocks) will be left open to the implementer of the standard.

This standard provides specific focus on the logical aspects of the SOA RA, and provides a model for including key architectural considerations and making architectural decisions through the elements of the meta-model.

Three of the layers address the implementation and interface with a service (the Operational Systems Layer, the Service Component Layer, and the Services Layer). Three of them support the consumption of services (the Business Process Layer, the Consumer Layer, and the Integration Layer). Four of them support cross-cutting concerns of a more supporting (sometimes called non-functional or supplemental) nature (the Information Layer, the Quality of Service Layer, the Integration Layer, and the Governance Layer). The SOA RA as a whole provides the framework for the support of all the elements of an SOA, including all the components that support services and their interactions.

This logical view of the SOA RA addresses the question: “If I build an SOA, what would it look like, how is it structured, and what abstractions should be present?” Also, for the assessment usage scenario of the SOA RA, the question answered by this standard is, informally: “If I assess an architecture proposing to be built upon SOA principles, what considerations and building blocks should be present and what shall we assess against?”

The SOA RA enumerates the fundamental elements of an SOA solution or enterprise architecture standard for solutions and provides the architectural foundation for the solution.

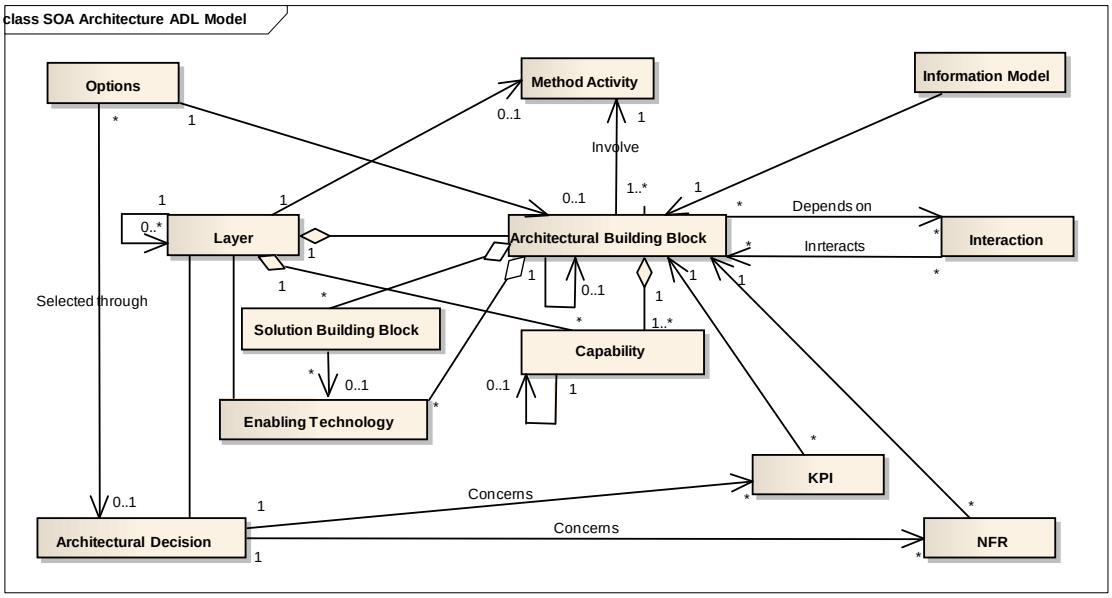


Figure 3: Meta-Model for Instantiating the SOA RA for a Given Solution

As shown in Figure 3, the meta-model of the SOA RA includes the following elements:

- *Layer*: An abstraction of a grouping of a cohesive set of ABBs, architectural decisions, interactions among ABBs, and interactions among layers, that support a set of related capabilities.
- *Capability*: An ability that an organization, person, or system possesses to deliver a product or service. A capability represents a requirement or category of requirements that fulfill a strongly cohesive set of needs. This cohesive set of needs or functionality is summarized by name given to the capability.
- *ABB (Architecture Building Block)*: A constituent of the architecture model that describes a single logical aspect of the overall model [10]. Each layer can be thought to contain a set of ABBs that define the key responsibilities of that layer. In addition, ABBs are connected to one another across layers and thus provide a natural definition of the association between layers. The particular connection between ABBs that recur consistently in order to solve certain classes of problems can be thought of as patterns of ABBs. These patterns will consist not only of a static configuration which represents the cardinality of the relationship between building blocks, but also the valid interaction sequences between the ABBs. In this SOA RA, each ABB resides in a layer, supports capabilities, and has responsibilities. It contains attributes, dependencies, constraints, and relationships with other ABBs in the same layer or different layer.
- *Method Activity*: A set of steps that involve the definition or design associated with ABBs within a given layer. The method activity provides a dynamic view of how different ABBs within a layer will interact. Method activities can also be used to describe the interaction between ABBs across the layers, so that an entire interaction from a service invocation to service consumption is addressed.

- *Options*: A collection of possible choices available in each layer that impact other artifacts of a layer. Options are the basis for architectural decisions within and between layers, and have concrete standards, protocols, and potentially solutions associated with them. An example of an option would be choosing SOAP or REST-style SOA services since they are both viable options. The selected option leads to an architectural decision.
- *Architectural Decision*: A decision derived from the options. The architectural decision is driven by architectural requirements, and involves governance rules and standards, ABBs, KPIs, and Non-Functional Requirements (NFRs) to decide on standards and protocols to realize an instance of a particular logical ABB. This can be extended, based on the instantiation of the SOA RA to the configuration and usage of ABBs. Existing architectural decisions can also be re-used by other layers or ABBs.
- *Interaction Pattern*: An abstraction of the various relationships between ABBs. This includes diagrams, patterns, pattern languages, and interaction protocols.
- *KPI (Key Performance Indicator*): A KPI may act as input to an architectural decision.
- *NFR (Non-Functional Requirement)*: An NFR may act as input to an architectural decision. NFRs help address Service-Level Agreement (SLA) attributes (e.g., response time, etc.) and architectural cross-cutting concerns such as security.
- *Enabling Technology*: A technical realization or instance of ABBs in a specific layer. Examples are web services or REST.
- *Information Model:* A structural model of the information associated with ABBs including information exchange between layers and external services. The information model includes the metadata about the information being exchanged.
- *Solution Building Block:* A runtime realization or instance of ABBs in a specific layer. A candidate physical solution for an ABB; e.g., a Commercial Off-The-Shelf (COTS) package such as a particular application server.

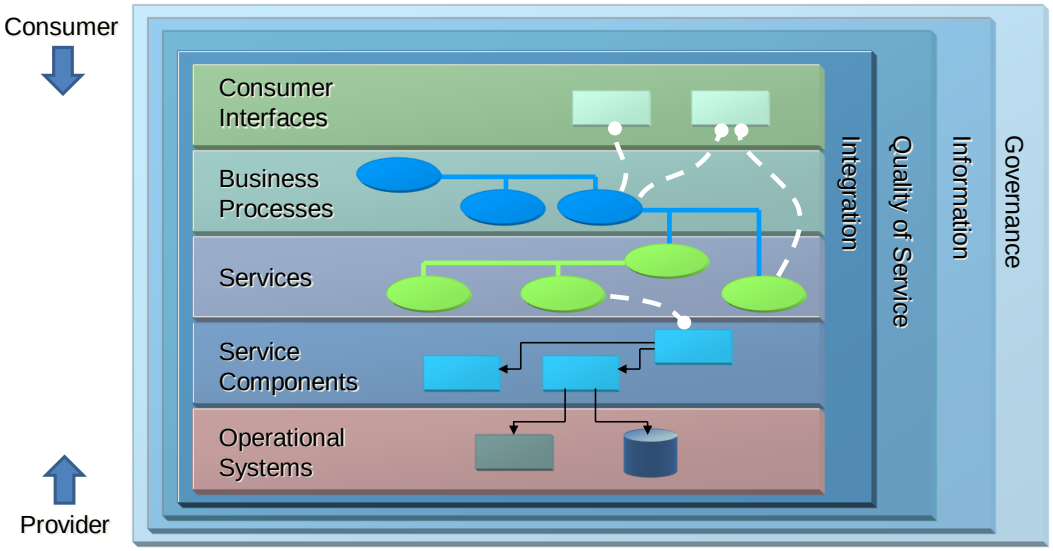


Figure 4: Logical Solution View of the SOA RA

Figure 4 depicts an SOA as a set of logical layers. Note that one layer does not solely depend upon the layer below it and is thus named a partially-layered architecture: a consumer can access the Business Process Layer or the Services Layer directly, but not beyond the constraints of an SOA architectural style. For example, a given SOA solution may exclude a Business Process Layer and have the Consumer Layer interacting directly with the Services Layer. Such a solution would not benefit from the business value proposition associated with the Business Process Layer; however, that value could be achieved at a later stage by adding the layer. In this sense the SOA RA represents SOA with a partially layered architecture. The degree to which a given organization realizes the full SOA RA will differ according to the level of SOA maturity they exhibit, and the underlying requirements of the organization.

Figure 4 illustrates the multiple separations of concern in the nine layers of the SOA RA. The SOA RA does not assume that the provider and the consumer are in one organization, and supports both SOA within the enterprise as well as across multiple enterprises in the industry ecosystem. The need for both intra and inter-enterprise SOA is important, with the role of SOA as the foundation of Cloud Computing and Software as a Service (SaaS). The Services Layer is the decoupling layer between consumer and provider.

The lower layers (Services Layer, Service Component Layer, and Operational Systems Layer) are concerns for the provider and the upper ones (Services Layer, Business Process Layer, and Consumer Layer) are concerns for the consumer. The main point of the provider/consumer separation is that there is value in decoupling one from the other along the lines of business relationship. Organizations which may have different lines of business use this architectural style (where one is the consumer and the other is the provider), customizing it for their own needs and integrating and interacting among themselves; in such a case there is still real value in maintaining a decoupled consumer/provider relationship. Below we will describe each layer and describe the relationships between the layers in the subsequent sections.

Note that there are five horizontal layers that are more functional in nature and relate to the functionality of the SOA solution. The supporting layers are supportive of cross-cutting concerns that span the functional layers but are clustered around independent notions themselves as cross-cutting concerns of the SOA architectural style.

6 Capabilities and the SOA RA

A capability, as defined by The Open Group, is: "*an ability that an organization, person, or system possesses*" [10]. From a TOGAF context: capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve. Marketing, customer contact, outbound telemarketing, etc. are illustrative examples for capabilities. The term capability can represent pure business capability such as Process Claim or Provision Service Request and can also represent technical capability such as Service Mediation or Content-Based Routing. Both business and technical capabilities are represented in SOA and enabled and supported by SOA. Using a capability modeling as part of the approach has some major advantages:

1. Allows us to focus on the "what" rather than the "how". This supports an abstract approach that is focused on the requirements of the solution.
2. Enables business capabilities to be aligned to the technical capabilities required to service them. Through the use of the SOA RA the associated enterprise-level and solution architectures can then be derived.
3. Enables us to derive and re-balance the SOA roadmap in an agile fashion. For example, if an organization foresees the need for integrating services across different business units, it might require a certain set of SOA layers and Architecture Building Blocks (ABBs) to be enabled.

The capability mapping process itself is out of scope of this document, and is normally a part of the organizational service modeling methodologies. The ability to derive the solution architecture from the SOA RA itself is within the scope of the SOA RA.

For example, the business capability to cross-sell requires a technical capability to have a common shareable set of data, where the data is from different systems in an enterprise. This in turn requires shareable metadata about data, supporting "information services" in some form and the ability to transport, mediate, and share data from the disparate systems in a common "enterprise" form. Thus, a business capability (cross-selling) is dependent on technical capabilities (the need to be able to have a common view of data (information services), the need to mediate, integrate, and transport the data, etc.). Each of these capabilities maps to ABBs supported by the layers of the SOA RA.

A capability-based approach enables us to answer the question: "When do we need a particular SOA RA layer?" and to help facilitate making decisions when organizational priorities change. The SOA RA further helps us by determining whether there are inter-dependencies and technical requirements for a layer and its constituent building blocks, beyond those defined by business capabilities, to create a holistic set of capabilities which the SOA RA needs to satisfy.

Figure 5 below shows the relationships among the core constructs of the SOA RA.

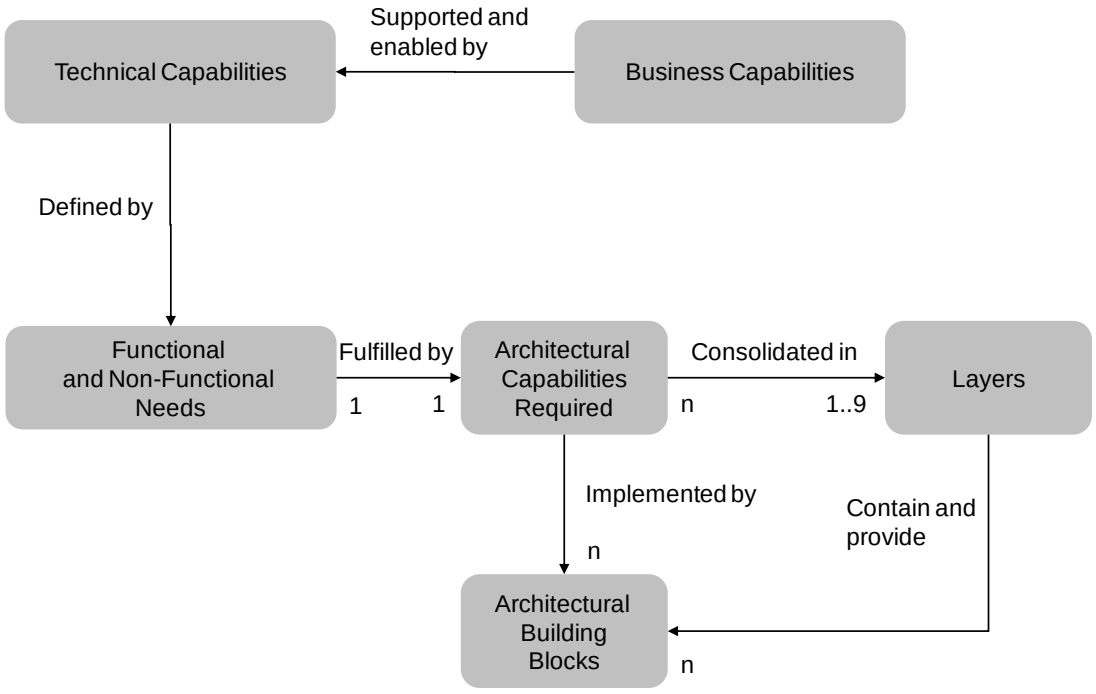


Figure 5: Relationships among Requirements, Capabilities, Building Blocks, and Layers

The layers in the SOA RA provide a convenient means of consolidating and categorizing the various capabilities and building blocks that are required to implement a given SOA. Below we will explore the details of these layers and their constituent elements.

7 Description of Layers

7.1 Assumptions

A Service-Oriented Architecture (SOA) is defined by the set of functional and Non-Functional Requirements (NFRs) that constrain it. Functional requirements are business capabilities imperative for business operations including business processes, the business and IT services, the components, and underlying systems that implement those services. NFRs for SOA include: security, availability, reliability, manageability, scalability, latency, governance and integration capabilities, etc.

The underlying requirements which determine the capabilities that the SOA supports are determined by:

1. A set of service requirements which includes business (*aka* functional) and NFRs on a service.
2. Service requirements result in the documented capability that a service needs to deliver or is expected to deliver.
3. The provider view of a service requirement is the business and technical capability that a given service needs to deliver given the context of all of its consumers.
4. The consumer view of a service requirement is the business and technical capability that the service is expected to deliver in the context of that consumer alone.

The fulfillment of any service requirement may be achieved through the capabilities of a combination of one or more layers in the SOA Reference Architecture (SOA RA).

Services themselves have a contract element and a functional element. The service contract or service interface defines what the service does for consumers, while the functional element implements what a service is obligated to provide based on the service contract or service interface. The service contract is integrated with the underlying functional element through a component which provides a binding. This model addresses services exposing capabilities implemented through legacy assets, new assets, services composed from other services, or infrastructure services.

The identification of service requirements and the mapping of those requirements to each of the layers of the SOA RA is a key aspect in developing an SOA for an enterprise [2][3].

In order to describe each of the layers of the SOA RA we need the following for each layer:

1. Introduction: Provide an overview of the layer itself.
2. Requirements: Provide an understanding of the capabilities supported by the layer and what they are (the answer to the “what does the layer do” question).

3. Logical Aspect: Provide an overview of the structural elements of the layer, applying the meta-model.

4. Interaction: Provide typical interactions among the Architecture Building Blocks (ABBs) within the layer and across layers.

In general, we follow a theme where each layer has a part which supports a set of capabilities/ABBs which support the interaction of the layer with other elements in the SOA RA; a part which supports the actual capabilities that the layer must satisfy; and a part which supports the orchestration and management of the other ABBs to support the layer's dynamic, runtime existence. Thus, in the following chapters that describe the layers in greater detail, we:

- Provide an overview and description of the layer and motivation behind the layer
- Provide the key capabilities supported by the layer
- Provide a structural overview of the layer which includes detailed description of ABBs enabling the responsibilities of the layer
- Describe the interactions within the layer and across other layers in the SOA RA

Figure 6 describes an interaction between the Consumer Layer and the Business Process Layer using the Integration Layer.

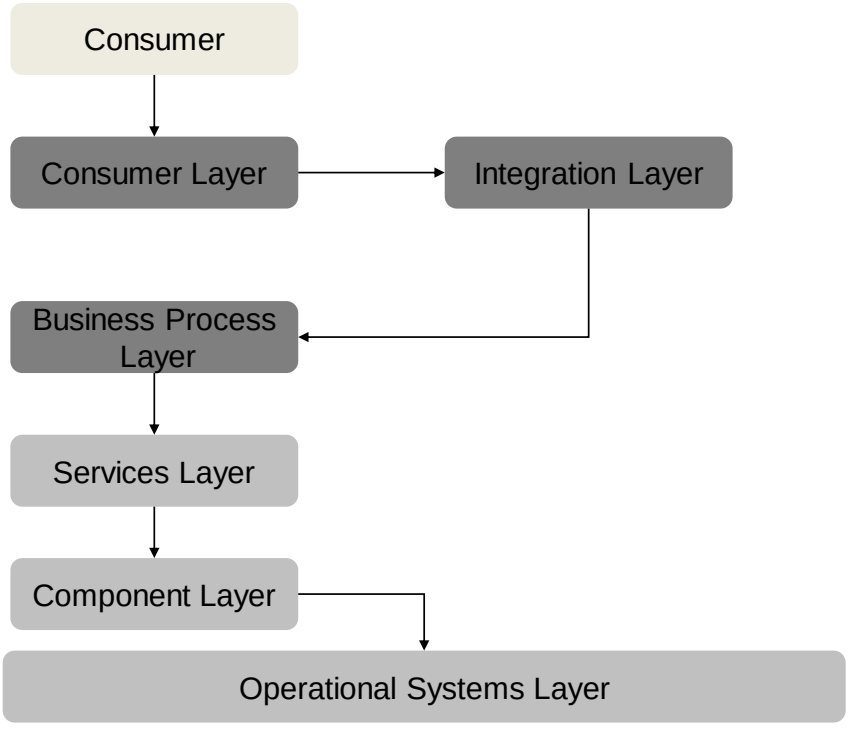


Figure 6: Typical Interactions among the Layers of the SOA RA

A typical interaction flow among the layers of the SOA RA is described below:

1. Service consumers request services using the Integration Layer.

2. The Integration Layer invokes the business process in the Business Process Layer which is using one or more services.
3. It invokes the Services Layer.
4. The Services Layer binds and invokes Service Components in the Service Component Layer.
5. Service Components in the Service Component Layer invoke Solution Components from the Operational Systems Layer to carry out the service request.
6. The response is sent back up to the service consumer.

8 Operational Systems Layer

8.1 Overview

All runtime elements of architecture reside in this layer. Effectively, this layer can conceptually be thought of as the runtime or deployment time of the solution. If we conduct a thought experiment and "freeze" the operations within this layer in a frame of time and expand it out, we will discover the separation of concerns that tend to cluster building blocks within the architecture: the aspects most closely connected with the consumption of the services, the processes that are choreographed into a flow, the services whose interfaces are exposed for consumption, the Service Components that will ultimately be used to realize the implementation of the services, along with the other four major cross-cutting and supporting concerns (integration, information, Quality of Service (QoS) factors, and governance).

8.1.1 Context and Typical Flow

This layer describes the runtime and deployment infrastructure; the programs, platforms, application servers, containers, runtime environments, packaged applications, virtual machines, etc. that are on the hardware and are needed to support the SOA solution. These specifically include:

- All software and hardware infrastructure necessary to support the SOA and its components at runtime and design time (tools)
- All operational and runtime hosting of underlying physical system components
- All assets required to support the functionality of services in the SOA, including custom or packaged application assets, new services, services created through composition or orchestration, infrastructure services, etc.

This layer represents a "snapshot" and logical categorization and generalization of the runtime environment. As such, this layer supports all the capabilities of the infrastructure that are needed for running/executing all software. Therefore, this layer supports the execution of the capabilities and responsibilities of the other layers of the SOA RA, including the components implementing the services; i.e., those components that a service relies on for providing it with its functional capabilities.

For example, if we have a capability for an SOA that involves systems using mainframe and J2EE platforms, the Operational Systems Layer would instantiate the necessary Architecture Building Blocks (ABBs) in the Integration Layer and Service Component Layer and the underlying mainframe and J2EE components which provide the functional capability.

We can express this as a formula such as:

Operational Systems Layer = [Infrastructural elements of all other layers] + [Underlying infrastructure to run the infrastructural elements (i.e., Operating Systems, etc.)] + [Elements that realize the Functional Components of services]

Thus this layer provides the building blocks supporting the operational systems which implement the functional capabilities of the other horizontal layers and the supporting/cross-cutting layers. In particular, the capabilities supported by this layer include providing operational and runtime hosting, infrastructure services and infrastructure virtualization, functional delivery support including support for service implementations, and realizations.

Connection Point with Other Initiatives or Standards

This layer represents the intersection point between the actual runtime infrastructure and the rest of the SOA which runs on that infrastructure. In addition, it is the integration point for an underlying Infrastructure as a Service (IaaS) construct and the rest of the SOA in the wider context of cloud computing. Key requirements for this layer are outlined in the capability section describing the capabilities provided to fulfill those requirements.

8.1.2 Capabilities

There are multiple categories of capabilities that the Operational Systems Layer needs to support. These categories of capabilities are:

- **Service Delivery**: This category of capabilities is required for delivery of the functional elements of services. This includes the finding of the components implementing the services, the wrapping and the composition/ decomposition of the underlying services, and the implementation of the services.
- **Runtime Environment**: This category of capabilities is required for providing a runtime environment representing runtime infrastructure for SOA. This includes capabilities to support both the components required to support service functionality and those required to actually run the components and building blocks of the SOA RA itself. This includes capabilities for:
  - — The hardware, operating system components
  - — The Solution Building Blocks, which are the runtime instances or realizations of the ABBs of all layers in the SOA RA that have been selected for inclusion in a particular operating environment.
- **Virtualization and Infrastructure Services**: This category of capabilities provides underlying infrastructure such as computing power, network, storage, etc. in native or a virtualized manner.

This layer features the following capabilities:

Service Delivery

1. Ability to locate components implementing services

2. Ability to host applications and functionality to deliver service features
3. Ability to host databases needed for service implementation
4. Ability to host legacy systems needed for service implementation
5. Ability to act as a broker between services requests and invoking implementations
6. Ability to map service functional requirements to underlying or legacy solution
7. Ability to compose service function from underlying services and implementation of services
8. Ability to wrap custom and legacy platforms for service implementation
9. Ability to find Service Component associated with Solution Building Blocks
10. Ability to delegate request or invoke Solution Component for service

Runtime Environment

11. Ability to support operating systems platforms
12. Ability to support runtime hosting platforms
13. Ability to support runtimes of software needed to run service implementation
14. Ability to support runtimes and software needed to deploy service implementations
15. Ability to run supporting ABBs and Solution Building Blocks from other layers of the SOA RA
16. Ability to support the software environment in which the Solution Component runs

Virtualization and Infrastructure Services

17. Ability to provide infrastructure needed by runtime infrastructure
18. Ability to provide infrastructure in a virtualized manner to platforms
19. Ability to provide infrastructure in a virtualized manner to service implementation
20. Ability to manage infrastructure and virtualized infrastructure
21. Ability to provide a single point of control for security of the Operational Systems Layer

8.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Operational Systems Layer are:

#	Capability Category	ABB Name	Supported Capabilities
	Service Delivery	Solution Component	1-4
		Implementation Controller	5-10

#	Capability Category	ABB Name	Supported Capabilities
		Integration Layer: Integration Controller	5
		Applications (Packaged and Custom)	2, 4
		Legacy System	4
		Database	3
		Quality of Service Layer: Policy Enforcer	
		Quality of Service Layer: Access Controller	2
	Runtime Environment	Runtime Hosting Environment	11-13
		Solution Platform	12
		Solution Building Block	15-16
		Deployment Unit	14
	Virtualization and Infrastructure Services	Hardware	17
		Virtualized Infrastructure	18-19
		Quality of Service Layer: IT Systems Manager	20
		Quality of Service Layer: Security Manager	21

Table 1: ABB to Capability Mapping for the Operational Systems Layer

8.2 Details of ABBs and Supported Capabilities

8.2.1 Details of ABBs

8.2.1.1 Solution Component

This ABB represents realization of subsystems that represent logical groupings of functionality and associated functionally cohesive services. Its bill of material contains the Service Component, Functional Components, and Technical Components from the Service Component Layer realizing services that provide well-defined interfaces for the subsystems. For example, it could be an invocation of a legacy component, or an existing or new database request, application, or a component wrapped within a Commercial Off-The-Shelf (COTS) package. Thus, it is a runtime instantiation of a group of cohesive components residing within a subsystem that collectively provide an implementation for a set of related services.

8.2.1.2 Implementation Controller [IC]

This ABB receives a request to invoke an underlying Solution Component and delegates it to the appropriate Solution Component. It also incorporates logic for composition and decomposition of legacy applications into Solution Components. This is required because historically most legacy applications have not been written with the intent of being elements in an SOA and the service Solution Components within them need to be exposed through service composition and decomposition.

8.2.1.3 Integration Layer: Integration Controller

This ABB is responsible for coordinating and brokering or mediating the interactions between the applications, database, security, etc. that need to work in concert to effectively provide a runtime experience.

8.2.1.4 Applications (Packaged and Custom)

This ABB represents the applications that are running as units of execution within the runtime environment.

8.2.1.5 Legacy System

This ABB describes the legacy systems running in the Operational Systems Layer.

8.2.1.6 Database

This ABB represents the databases running in the Operational Systems Layer.

8.2.1.7 Quality of Service Layer: Policy Enforcer

See Policy Enforcer ABB in the Quality of Service Layer.

8.2.1.8 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

8.2.1.9 Runtime Hosting Environment [RHE]

This ABB provides support for operational and runtime services. These include software services such as the operating system instance on which the Solution Platforms run, as well as underlying infrastructural services such as hardware support, memory, storage, networks, etc.

8.2.1.10 Solution Platform

This ABB supports the software environment in which the Solution Components and Solution Building Blocks deploy and run. Examples would be Java Virtual Machines (JVMs) hosting a Service Container Solution Building Block, or a CICS environment.

8.2.1.11 Solution Building Block

This ABB represents the runtime component of ABBs from other layers in the SOA RA. Thus, for example, a Mediation ABB from the Integration Layer runs as a Solution Building Block.

8.2.1.12 Deployment Unit

This ABB represents an executable application that can be deployed as a single unit (e.g., exe, war, ear, etc.) in the target hosting environment. This ABB is deployed on Solution Platforms.

8.2.1.13 Hardware

This ABB represents an abstraction of the physical hardware that is the platform on which Deployment Units are actually deployed and are executing (running).

8.2.1.14 Virtualized Infrastructure

This ABB supports the utilization of infrastructure in a virtualized manner by the operational and hosting runtime environment. Thus, the use of shared disk space in a cloud environment would be an example.

8.2.1.15 Quality of Service Layer: IT Systems Manager

See IT Systems Manager ABB in the Quality of Service Layer.

8.2.1.16 Quality of Service Layer: Security Manager

See Security Manager ABB in the Quality of Service Layer.

8.2.2 Structural Overview of the Layer

The ABBs in the Operational Systems Layer can be thought of as being logically partitioned into the categories that support:

- Solution Components which provide the functional capability of services and the solution
- Runtime environment required by the SOA and that runs the actual Solution Components and their supporting infrastructure
- Supports the interfacing with underlying infrastructure, so that they can be virtualized and leveraged as such by the underlying architecture

In the Structural Overview of the Layer diagrams in this specification, the ABBs are color-coded to match the other layers in the architecture that they belong to. For example, in the following diagram the ABBs from the Quality of Service Layer are a dark gray, while the ABBs from the Integration Layer are a black.

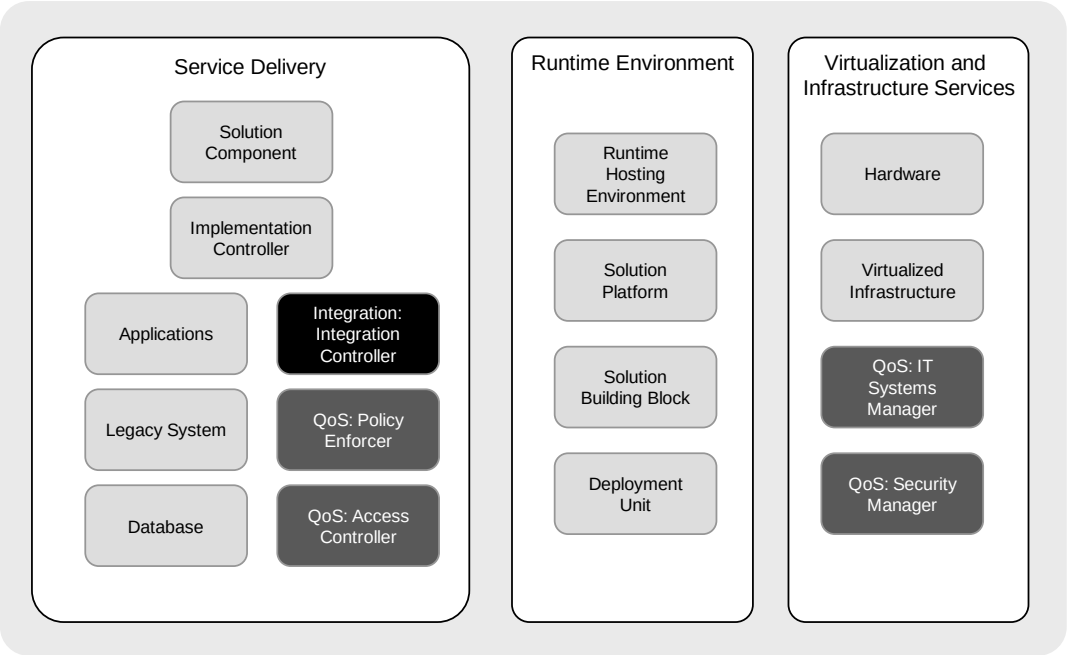


Figure 7: ABBs in the Operational Systems Layer

8.3 Inter-Relationships between the ABBs

The Solution Component ABB represents realization of subsystems that represent logical grouping of functionality and associated functionally cohesive services. Its bill of material contains the Service Component, Functional Components, and Technical Components from the Service Component Layer realizing services providing well-defined interfaces for the subsystems. Requests are first validated as secure by the Access Controller ABB and Security Manager ABB in the Quality of Service Layer, and then translated into Solution Component ABB requests by the Implementation Controller ABB and executed by the Solution Components in the Solution Platform.

The runtime hosting and operational environment capability is supported by the Solution Platform ABB and Runtime Hosting Environment ABB. Thus, both ABBs from all layers of the SOA RA run as Solution Building Blocks on the Solution Platform hosted by the Runtime Hosting Environments.

The infrastructure services and infrastructure virtualization capability basically is responsible for exposing the underlying infrastructure in an on-demand manner, encapsulating the invoking ORE and enabling rapid scaling.

In the following diagram, the arrows between the ABBs indicate an interaction from one ABB to another.

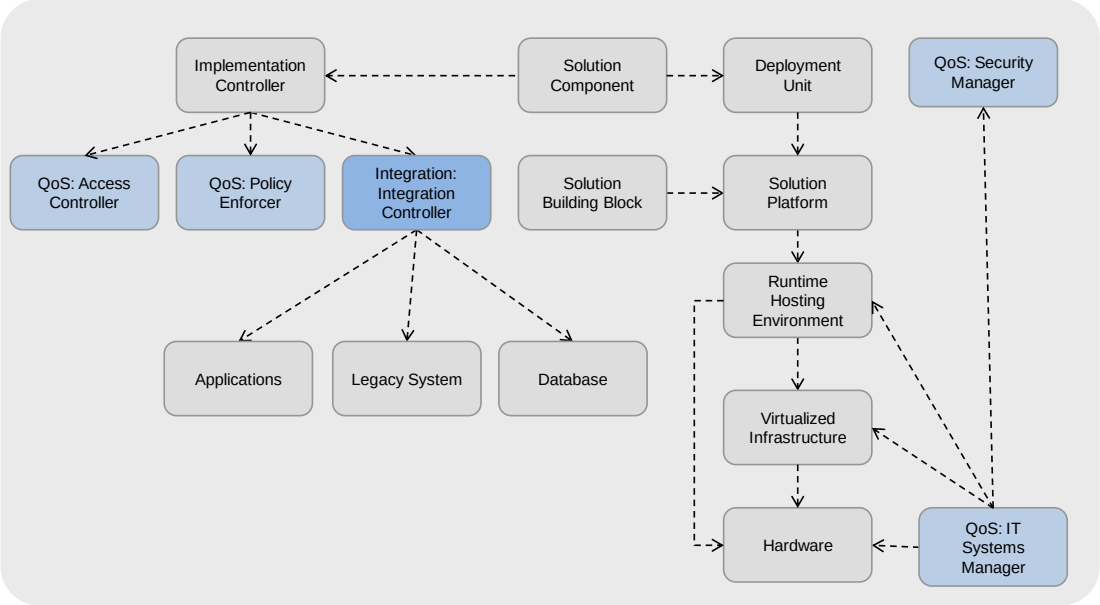


Figure 8: Relationships among ABBs in the Operational Systems Layer

8.4 Significant Intersection Points with other Layers

There is a significant intersection between the Operational Systems Layer and all other layers of the SOA RA as this layer provides the actual runtime environment to the other layers to execute at runtime. It intersects with the rest of the SOA RA including, of course, the cross-cutting and supporting layers, such as the Quality of Service Layer which acts as an aggregation point for monitoring, managing, and securing the runtime environment.

We will summarize this connection below in terms of intersection with the rest of the SOA RA including cross-cutting or supporting layers.

8.4.1 Intersection with the Rest of the SOA RA

1. There are two points of intersection of the Operational Systems Layer with the rest of the SOA RA: The first perspective is that the Operational Systems Layer supports the functionality of the SOA RA rendered by the more abstract layers above. There are two parts to this perspective: providing a wrapper which exposes the service aspect from the Service Component Layer, providing a mapping to the underlying Solution Component ABB which actually supports the capability. The Solution Platform ABB provides a platform to deploy and run the Solution Components realizing the functionally cohesive services in a subsystem.

2. The second perspective is that the Operational Systems Layer provides the runtime environment for the ABB from other layers. These ABBs from the other layers are instantiated as Solution Building Blocks for the runtime environment. The Solution Platform ABB provides a platform to deploy and run these ABBs from other layers.

8.4.2 Interaction with Cross-Cutting Layers

The Operational Systems Layer relies on cross-cutting layers of the architecture to fulfill its responsibilities.

It relies on the Quality of Service Layer for the following capabilities:

- Ability to authenticate/authorize for service invocation
- Ability to enforce operational policies
- Ability to monitor health and wellbeing of underlying infrastructure and solution and applications deployed on the infrastructure

It relies on the Information Layer for the following capabilities:

- Ability to store and retrieve metadata and data

It relies on the Integration Layer for the following capabilities:

- Ability to invoke business processes and/or services
- Ability to transform data from one format to another

It relies on the Governance Layer for the following capabilities:

- Ability to set and store business rules
- Ability to manage policies for IT management
- Ability to manage security policies

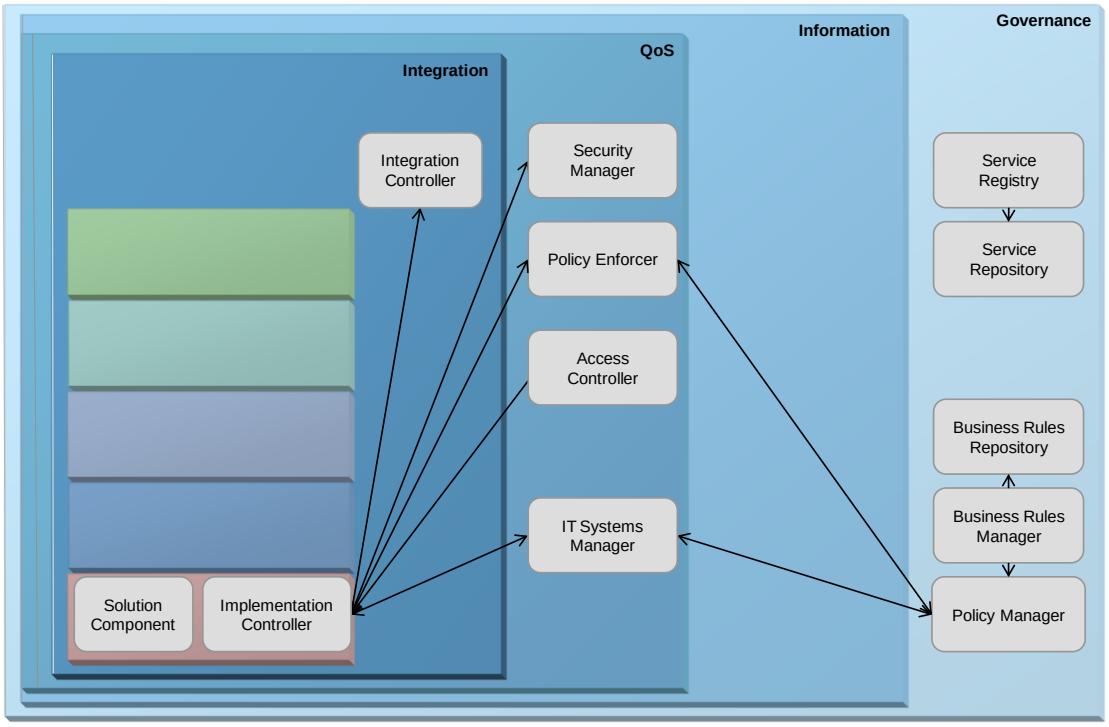


Figure 9: Key Interactions of Operational Systems Layer with Cross-Cutting Layers

Therefore, Operational Systems Layer interfaces with the following ABBs of cross-cutting layers of the architecture to provide its capabilities:

1. It leverages Policy Manager ABB in the Governance Layer enabling consolidation of policies as well as the management and administration of the security policies in one place – addressing the very important issue of security in the distributed environment as in the case of an SOA. It should be noted that in practice it may coordinate or integrate with the security mechanisms of the Solution Platform and Runtime Hosting Environment in which the SOA runs.

2. It leverages Access Controller ABB in the Quality of Service Layer to enforce access privileges and Policy Enforcer ABB in the Quality of Service Layer to enforce policies. These ABBs from the Quality of Service Layer enable the Operational Systems Layer to operate across platforms and support a consistent set of policies for specific scenarios, and limit the amount of associated risk. The Access Controller ABB and Policy Enforcer ABB in the Quality of Service Layer provide a single Policy Enforcement Point (PEP) for control for security for the Operational Systems Layer, and in practice for all the runtime components of the SOA RA. The policy enforcement could be federated. The Security Manager ABB in the Quality of Service Layer implements a participating filter pattern, where in-bound requests are submitted to policy enforcement and then the appropriate delegation of security to the Solution Platforms and Runtime Hosting Environment occurs.

3. It leverages IT Systems Management-related ABBs in the Quality of Service Layer such as Systems Manager ABB, Network Manager ABB, Storage Manager ABB, and

Application Systems Manager ABB to monitor, check heartbeat, and manage the infrastructure, systems, and applications.

4. It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Layer for coordinating and brokering or meditating the interactions between the applications, database, security, etc. that need to work in concert to effectively provide a runtime experience.

8.4.3 Interaction with Horizontal Layers

The Operational Systems Layer provides the runtime environment for other horizontal layers that are more functional in nature. Each of the other horizontal layers – namely, Consumer Layer, Business Process Layer, Services Layer, Service Component Layer – have some functional ABBs and some supporting ABBs that are needed to provide a runtime environment for the functional aspects of the solution. In a nutshell, the functional ABBs of the solution get rendered into a Solution Component in the Operational Systems Layer during runtime, whereas supporting ABBs get rendered into Solution Building Blocks in the Operational Systems Layer during runtime.

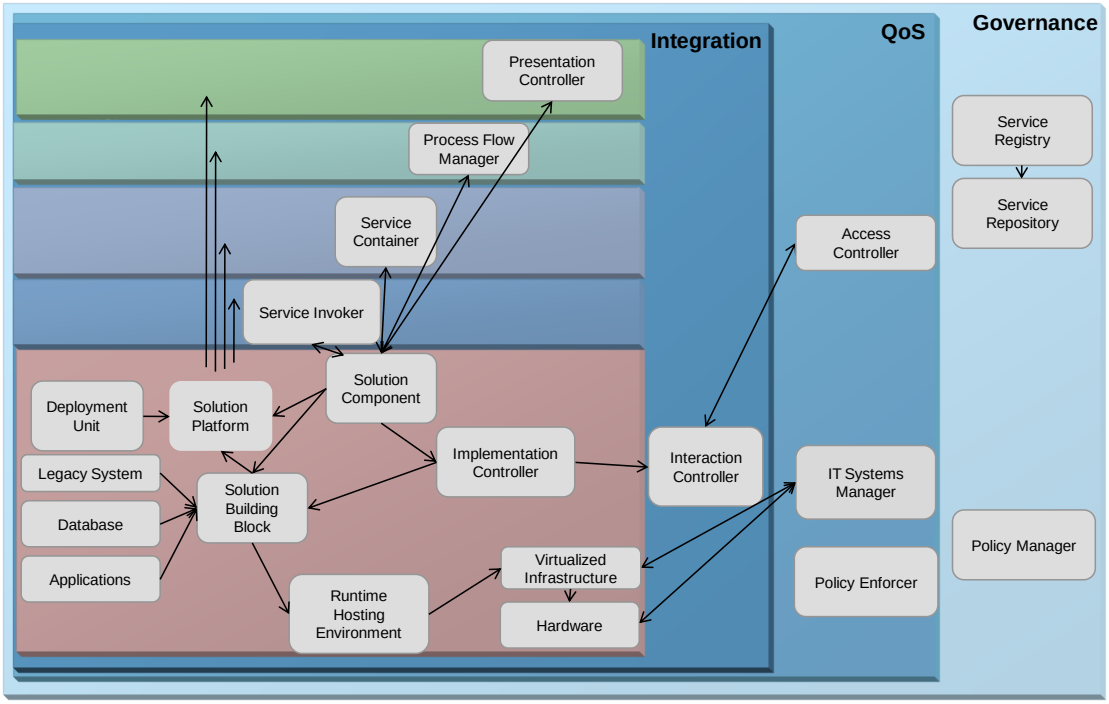


Figure 10: Key Interactions of Operational Systems Layer with Horizontal Layers

8.5 Usage Implications and Guidance

8.5.1 Options and Design Decisions

The capabilities supported by the Operational Systems Layer include enablement of infrastructure services for realizing the SOA; i.e., the (re-)use and composition of assets required as infrastructural elements for running the SOA.

From an SOA perspective, the Operational Systems Layer enables organizations to integrate in a perimeter-less, cross-organizational manner, such as Cloud-based virtualization in the form of SaaS applications which involves the integration of infrastructure services used in the Cloud, and the re-use of existing application assets originating from the diverse portfolio of custom and packaged applications that are running within an organization. This integration in a perimeter-less, cross-organizational fashion enables the foundation for service re-use by allowing the sharing of functionality and supporting capabilities across the portfolio.

In particular, this layer directly influences the overall cost of implementing SOA solutions within enterprises, the alignment and legacy modernization impact of SOA, the re-use of legacy solutions, and the positioning of SOA for next-generation SOA evolution, such as Cloud Computing.

Finally, it is important to note that a service executes its functionality through building blocks that are assets in this layer. For example, a patient record update service that contracts to update patient records does so using different components running in application assets hosted in the Operational Systems Layer.

A number of existing software systems are part of this layer. Those systems include but are not limited to:

- Existing monolithic custom applications including J2EE [17] and .NET [18] applications
- Existing SOA services
- Legacy applications and systems
- Existing transaction processing systems
- Existing databases
- Existing package applications and solutions including ERP and CRM packages

8.5.2 Implementation Considerations

Considerations when using this layer include:

- When dealing with legacy, custom, and COTS applications, plan on a layer to support composition/ decomposition and integrate with the underlying systems.
- Try to federate security, and potentially event monitoring, to support the traceability and the agility required for an effective SOA. For example, if you have a J2EE environment currently and build federation in, as you go through a Merger and Acquisition scenario

where the CICS, .NET, and SaaS components need to be added, a core security framework is critical for incorporating these components in an agile manner.

- When dealing with virtualized infrastructure, it is important to consider the following:

  — Isolation and containment services:

    - Multi-tenancy
    - Data privacy (both data in motion and at rest)
    - Auditability
    - Authorization/authentication/access control

  — Application support services:

    - Elasticity – dynamic resource provisioning/de-provisioning
    - Service location awareness
    - Infrastructure QoS management (as opposed to Application Service QoS)

  — Data integrity services:

    - Disaster recovery
    - High availability clustering
    - Data backup
    - Data QoS management
    - Data mobility

  — Infrastructure accounting services:

    - Chargeback services
    - Configuration management/auditability
    - Capacity management services

8.5.3 Runtime and Deployment View of the SOA RA

As described in the first paragraph, the Operational Systems Layer supports all the capabilities of the infrastructure that are needed for running/executing all software. Therefore, this layer supports the execution of the capabilities and responsibilities of the other layers of the SOA RA, including the components implementing a service itself and the components that provide the SOA capabilities like Service Container ABB from the Services Layer, Data Transformer ABB from the Integration Layer, Process Flow Manager ABB from the Business Process Layer, etc.

The Solution Building Block in the Operation Systems Layer is defined as representing the runtime component of ABBs from other layers in the SOA RA. Thus, for example, a Protocol Conversion ABB in the Integration Layer runs as a Solution Building Block in the Operational

Systems Layer. But a business service, such as Get Customer Information, runs ultimately as a Solution Component.

In the description of the various layers this infrastructural support from a deployment perspective will be mentioned several times. The service container component will be introduced in the Services Layer, but will actually contain/deploy components from both the Services Layer and the Service Component Layer. All runtime building blocks from the other layers will be deployed directly as Solution Building Blocks, as mentioned above.

In Figure 11 this deployment view of the SOA RA is visualized.

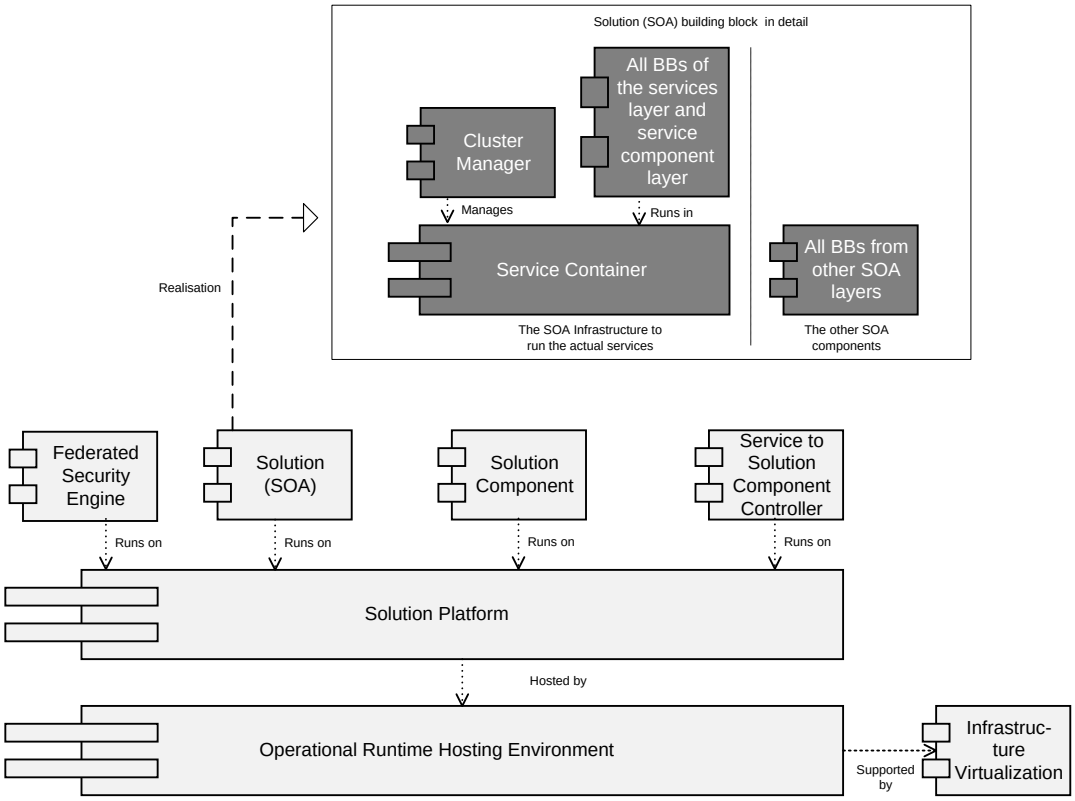


Figure 11: Deployment View of the SOA RA

The light gray colored building blocks are from the Operational Systems Layer. The dark gray colored building blocks are partly generalized building blocks from the other layers.

9 Service Component Layer

9.1 Overview

This layer contains software components, each of which provides the implementation or "realization" for services and their operations, hence the name "Service Component". The layer also contains the Functional and Technical Components that facilitate a Service Component to realize one or more services. Service Components reflect the definition of the service they represent, both in terms of functionality and Quality of Service (QoS). They "bind" the service contract/specification to the implementation of the service in the Operational Systems Layer. Service Components are hosted in containers which support the service specifications.

The Service Component Layer manifests the IT conformance with each service contract/description/specification defined in the Services Layer; it guarantees the alignment of IT implementation with service description.

In detail, each Service Component fulfils the following goals:

- Realizes one or more services
- Provides an enforcement point for service realization
- Enables IT flexibility by strengthening the decoupling in the system, by hiding volatile implementation details from service consumers

In particular, the Service Component Layer:

- Enables IT flexibility by strengthening the decoupling in the system; decoupling is achieved by hiding volatile implementation details from consumers
- Often employs container-based technologies, such as EJBs

Each Service Component:

- Realizes one or more services
- Provides an enforcement point for service realization
- Offers a façade behind which IT is free to do what is wanted/needed
- Generally contains business-specific logic with no reference to integration logic

9.1.1 Context and Typical Flow

The Service Component Layer provides the following:

- Ability to support the exposure of a service in a standards-compliant manner supporting interoperability; note that the protocol (SOAP/REST/J2EE, etc.) is not prescribed but determined by the associated architectural decision
- Ability to expose the service via an integration stack from the underlying platform in which the service functionality resides (*aka* within the Operational Systems Layer)
- Ability to publish and deploy the Service Component itself:
  - — Expose services in an interoperable manner
  - — Bind to the Operational Systems Layer at runtime
  - — Publish service contract information in an interoperable and standards-compliant manner so that other elements of the SOA can invoke it
  - — Deploy the service into the associated "services container"

9.1.2 Capabilities

There are multiple categories of capabilities that Service Component Layer need to support in the SOA RA. These capabilities include both design-time and runtime capabilities. These capability categories are:

- **Service Realization and Implementation**: This category of capabilities supports the realization of the services.
- **Service Publication and Exposure**: This category of capabilities supports service exposure and service contract publication.
- **Service Deployment**: This category of capabilities supports service deployment.
- **Service Invocation**: This category of capabilities supports service invocation.
- **Service Binding:** This category of capabilities supports service binding.

Note: Service Realization and Implementation, Service Publication and Exposure, and Service Deployment are design-time capabilities, while Service Invocation and Service Binding are runtime capabilities.

Service Realization and Implementation (Design Time)

1. Ability to realize a service; for example, using component-based design and development

Service Publication and Exposure (Design Time)

2. Ability to publish the service contract/descriptions in a standards-compliant, interoperable manner for other layers of the SOA RA and design-time service repositories and runtime service registry in the Governance Layer

3. Ability to provide information about the services to the Services Layer

Service Deployment (Design Time)

4. Ability to provide for the deployment of the physical service to the existing solution platform which contains the associated service Solution Component

Service Invocation (Runtime)

5. Ability to support a standards-compliant, interoperable, runtime invocation of the service

Service Binding (Runtime)

6. Ability to support service interoperability
7. Ability to implement a part of the broker pattern
8. Ability to convert from the service description to service calls supported by the platform (in the case of a WSDL web service, the conversion from WSDL service description to the service invocation)
9. Ability to convert at runtime into a standards-compliant form for consumption by standards-compliant service consumers on both input and output
10. Ability to convert from the standards-compliant form to the form acceptable to the underlying Solution Component which satisfies the service's functional capability on both input and output
11. Ability to enforce policies and access control during service binding

9.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Service Component Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Service Realization & Implementation	Service Component[2]	1
2		Functional Component	1
3		Technical Component	1
4	Service Publication and Exposure	Service Publisher	2-3
5		Integrated Development Environment (IDE) for Service Development	2
6		Governance Layer: Service Repository	2-3

[2] This can use any technology, not necessarily SCA.

#	Capability Category	ABB Name	Supported Capabilities
7	Service Deployment	Service Deployment Manager	4
8	Service Invocation	Service Invoker	5
9	Service Binding	Service Implementation Binder	6-8
10		Method Input/Output Transformer	9
11		Service Implementation Adapter	10
12		Quality of Service Layer: Policy Enforcer	11
13		Quality of Service Layer: Access Controller	11

Table 2: ABB to Capability Mapping for the Service Component Layer

These ABBs help in the realization of services and SOA in general and support the binding of services based on standards required for integration with other SOA RA layers in a standards-compliant and interoperable fashion.

9.2 Details of ABBs and Supported Capabilities

9.2.1 Details of ABBs

9.2.1.1 Service Component

This ABB realizes one or more services that are important to the enterprise to be managed and governed as an enterprise asset.

9.2.1.2 Functional Component

This ABB provides business functionality and aids in the realization of the Service Component. A Functional Component may be composed of other Functional Components and/or domain objects.

9.2.1.3 Technical Component

This ABB provides abstraction of infrastructure to support Functional Components.

9.2.1.4 Service Publisher

This ABB publishes Service Component design-time metadata and description to a design-time Service Repository ABB in the Governance Layer.

9.2.1.5 Integrated Development Environment (IDE) for Service Development

This ABB helps define and develop Service Components and associated Functional and Technical Components. This includes service contracts and any other associated service metadata.

9.2.1.6 Governance Layer: Service Repository

See Service Repository ABB in the Governance Layer.

9.2.1.7 Service Deployment Manager

This ABB deploys runtime Service Components to a services container and registers service description information in the Service Registry ABB in the Governance Layer. This can be automated through different mechanisms (from build scripts to an automated deployment, etc.).

9.2.1.8 Service Invoker

This ABB supports the invocation of the Service Components by the Services Layer. This includes invoking the Service Container to bind to and load the Service Component into the Service Container (a Services Layer ABB described in the Services Layer).

9.2.1.9 Service Implementation Binder

This ABB provides any bindings required to the invoking services and other layers. For example, if this were a WSDL-based service, then the invoking services or Integration Layer component would require its service request converted or mapped to an underlying Service Component call. The aspect of converting the in-bound call leverage Method Input/Output Transformer ABB is the responsibility of this ABB.

9.2.1.10 Method Input/Output Transformer

This ABB helps in transformation of input and output parameters of service operation and conversion of associated data elements from one format to another. It is used by the Service Implementation Adapter ABB to do the transformation/translation. It retrieves its metadata from the Information Layer and leverages the Data Transformer ABB in the Integration Layer to perform the necessary transformation.

9.2.1.11 Service Implementation Adapter

This ABB interfaces with the Operational Systems Layer and passes on the service call to the Operational Systems Layer in a form that is compliant with the Solutions Platform in the Operational Systems Layer, where the underlying Solution Components for the service are housed.

9.2.1.12 Quality of Service Layer: Policy Enforcer

See Policy Enforcer ABB in the Quality of Service Layer.

9.2.1.13 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

9.2.2 Structural Overview of the Layer

The Service Component Layer can be thought of as supporting capabilities dealing with design-time and runtime aspects. One of the key responsibilities of the Service Component Layer is to provide the integration between other SOA RA layers (e.g., the Integration Layer), and the

underlying Operational Systems Layer. The Service Component Layer thus supports the binding to other SOA RA layers and the standards required to support that interoperability. It also provides the binding to the Implementation Controller and thus the underlying Solution Building Blocks in the Operational Systems Layer. This binding is achieved through the implementation of the broker pattern.

The ABBs in the Service Component Layer can be thought of as being logically partitioned into the categories that support:

- Realization and implementation of services
- Exposure and contract publication of services
- Deployment of services
- Invocation of services
- Binding of services

Figure 12 illustrates the ABBs supporting the capabilities of the Service Component Layer.

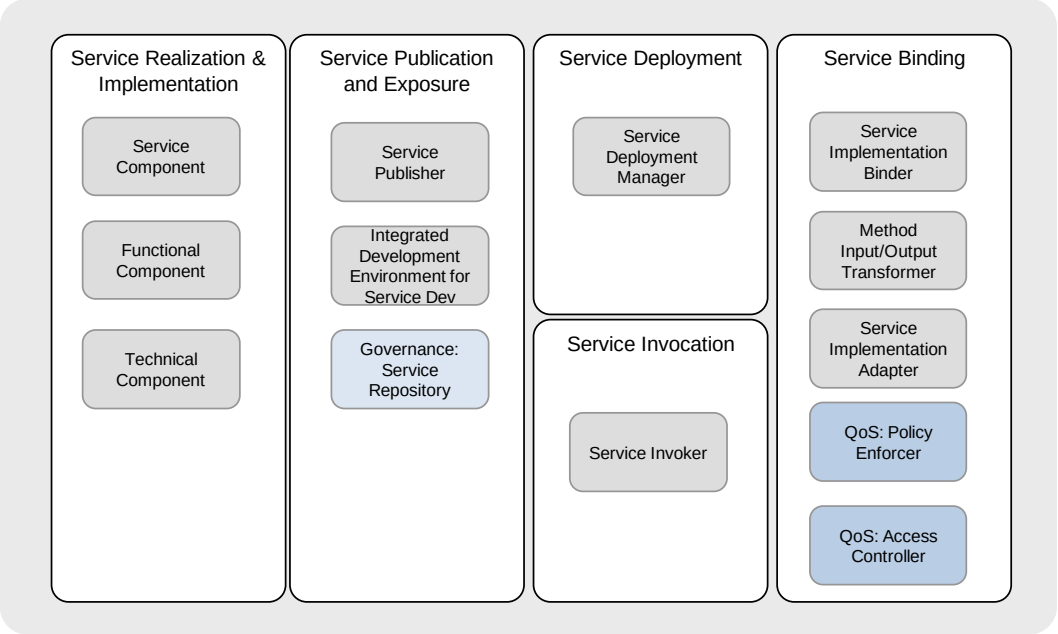


Figure 12: ABBs in the Service Component Layer

9.3 Inter-Relationships between the ABBs

As menitoned earlier, the ABBs in the Service Component Layer can be grouped as follows:

- ABBs that support the design-time capabilities of the layer
- ABBs that support the runtime capabilities of the layer

Figure 13 illustrates the interaction flows among ABBs in the Service Component Layer enabling design-time capabilities.

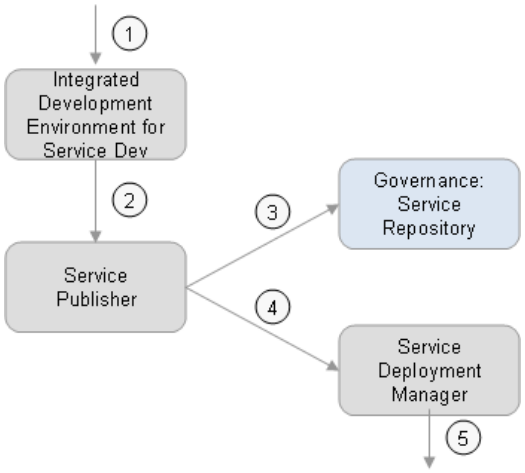


Figure 13: Illustrative Interaction Flow among Design-Time ABBs in the Service Component Layer

The interaction flow among design-time ABBs in the Service Component Layer is described as follows:

- During service design, an Integrated Development Environment (IDE) for Service Development is used to develop a service contract and specification using agreed upon standards.
- The service contract and specification is given to a Service Publisher ABB to publish the service.
- The Service Publisher ABB publishes the service contract and specification in the Service Repository ABB in the Governance Layer for use by other SOA layers and cross-cutting concerns.
- The Service Publisher also invokes a Service Deployment Manager to execute deployment functions so that the service is available to access in the appropriate services container, according to the contract and other information.

Figure 14 illustrates the interaction flows among ABBs in the Service Component Layer enabling runtime capabilities.

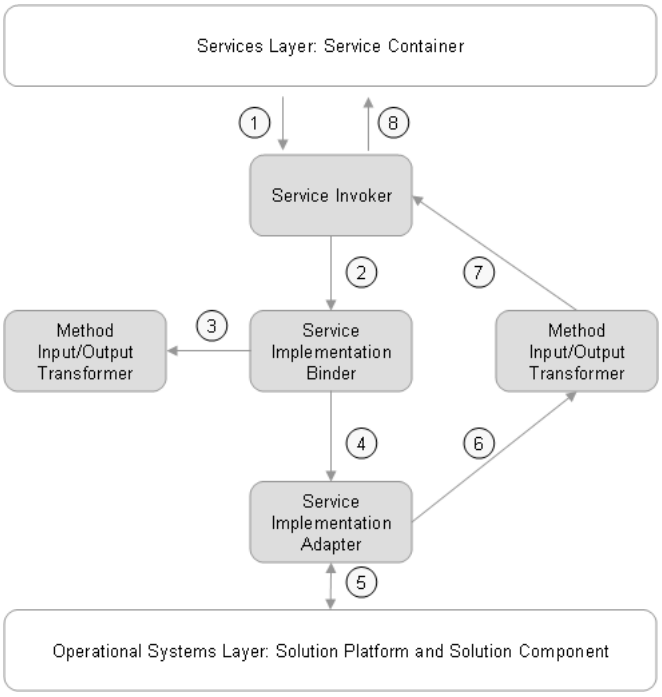


Figure 14: Illustrative Interaction Flow among Runtime ABBs in the Service Component Layer

The interaction flow among runtime ABBs in the Service Component Layer is described as follows:

- The Service Invoker ABB is invoked from all other layers of the SOA RA (except the Operational Systems Layer) and provides the ability for the Services Layer to invoke the Service Component realizing the services.
- The Service Invoker ABB calls the Service Component ABB which is the binding stack to bind to external layers.
- The Service Component ABB can invoke the Method Input/Output Transformer ABB to have the data formats transformed for interaction with the service consumers or other layers of the SOA RA.
- The Service Component ABB then passes control to the Service Implementation Adapter.
- The Service Implementation Adapter then maps the invocation into the Operational Systems Layer.

9.4 Significant Intersection Points with other Layers

The Service Component Layer provides the IT conformance with each service contract defined in the Services Layer and it guarantees the alignment of IT implementation deployed on the Operational Systems Layer with service description. Each Service Component:

- Provides an enforcement point for “faithful” service realization (ensures QoS and Service-Level Agreements (SLAs))

- Enables business flexibility by supporting the functional implementation of IT flexible services, their composition, and layering
- Enables IT flexibility by strengthening the decoupling in the system; decoupling is achieved by hiding volatile implementation details from consumers

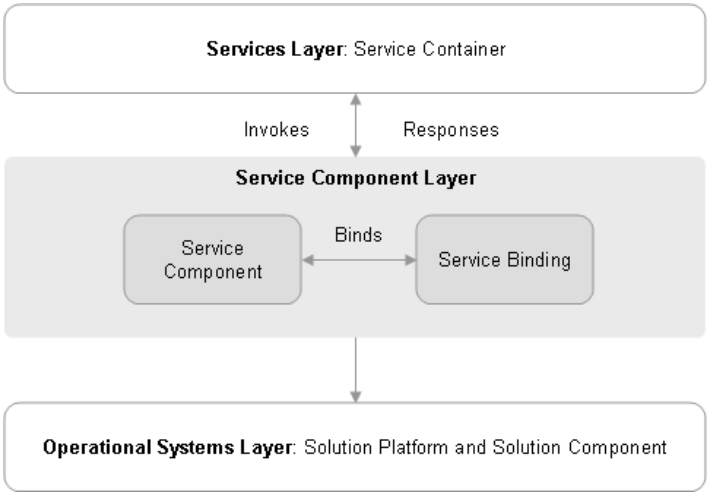


Figure 15: High-Level Interaction of the Service Component Layer with Layers Above and Below in the SOA RA

The Solution Component ABB in the Operational Systems Layer can be thought of as a runtime instantiation of a solution enabling a subsystem of services. A subsystem is implemented by one or more Service Components realizing one of more services and related Functional and Technical Components. The Solution Component ABB is a runtime instantiation of the Service Components and associated Functional and Technical Components realizing a subsystem of services. Service architects and developers must determine which standards to conform to in the protocols for the service as well as to connect to the underlying Operational Systems Layer. Which standards are used to describe the service (e.g., WSDL) and these protocols is also an important decision.

9.4.1 Interaction with Cross-Cutting Layers

The Service Component Layer relies on cross-cutting layers of the architecture to fulfill its responsibilities.

It relies on the Governance Layer for the following capabilities:

- Ability to store metadata about services during design time
- Ability to define and manage (storage, retrieval, etc.) of rules used by the components realizing the services

It relies on the Quality of Service Layer for the following capabilities:

- Ability to authorize during invocations on the underlying components.

It relies on the Information Layer for the following capabilities:

- Ability to store and retrieve metadata and data required by the components

It relies on the Integration Layer for the following capabilities:

- Ability to transform data from one format to another

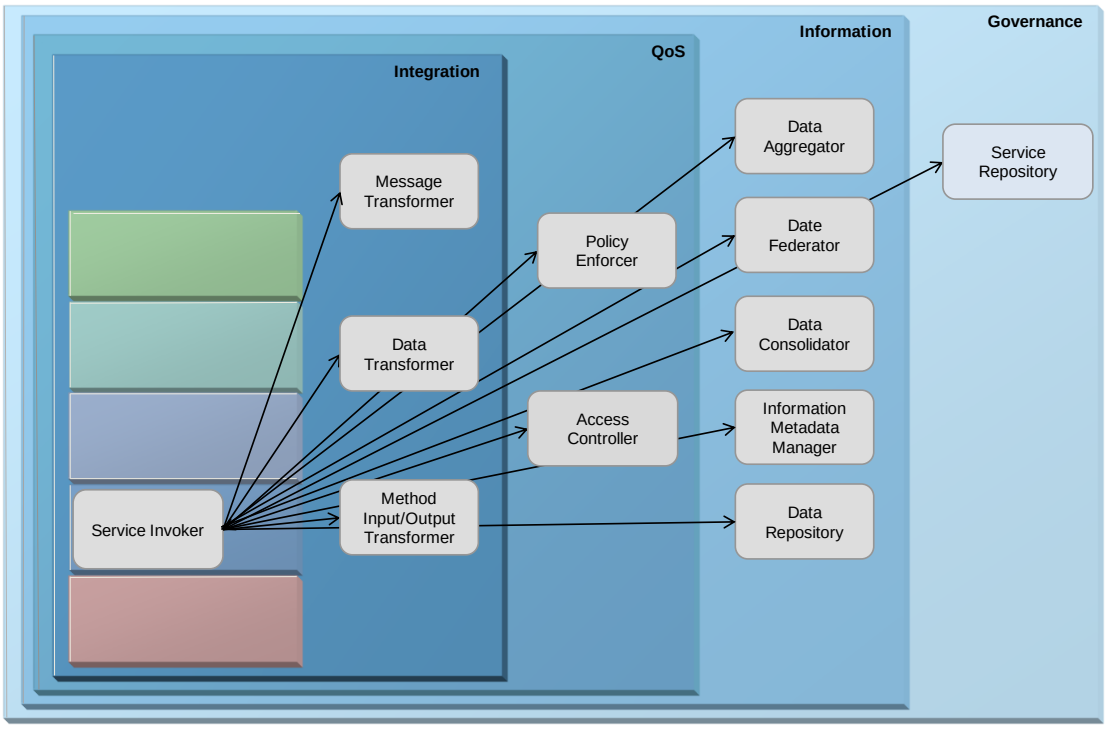


Figure 16: Key Interactions of the Service Component Layer with Cross-Cutting Layers

Therefore, the Service Component Layer interfaces with the following ABBs of cross-cutting layers of the architecture to provide its capabilities:

- It leverages the Access Controller ABB and Policy Enforcer ABBs in the Quality of Service Layer to enforce access control privileges and other policies.
- It leverages the Data Aggregator ABB, Data Federator ABB, Data Consolidator ABB, Information Metadata Manager ABB, and Data Repository ABB from the Information Layer to provide information about services for the other layers of the SOA RA.
- It leverages the Message Transformer ABB and Data Transformer ABB from the Integration Layer to transform data from one format to another. The Method Input/Output Transformer ABB leverages these ABBs from the Integration Layer.
- It leverages the Service Repository ABB in the Governance Layer to store metadata about services. IT Governance has a significant influence on the Service Component Layer. The choice of implementation technology, the manner in which Service Components may/may not consume behaviors from other Service Components, and decisions about where to place integration logic are examples of where this layer may be influenced by IT Governance and the Governance Layer. For another example, the implementation options for a Service Component may include BPEL, a Session EJB, a Message Broker Flow, a

SOAP/CICS operation, etc. Some of these alternatives may eliminate/involve a governance exception, because the Technology Roadmap established by IT Governance excludes them.

9.4.2 Interaction with Horizontal Layers

The Service Component Layer realizes the services from the Services Layer and then uses the Operational Systems Layer to execute the services in a runtime environment. In order to fulfill these core responsibilities, the ABBs in the Service Component Layer interact with the Services Layer and Operational Systems Layer.

- The Service Invoker ABB interacts with the Services Layer and Integration Layer.
- The Service Publisher ABB interacts with the Services Layer.
- The Service Deployment Manager ABB interacts with the Operational Systems Layer.
- The Service Implementation Adapter ABB interacts with the Operational Systems Layer.

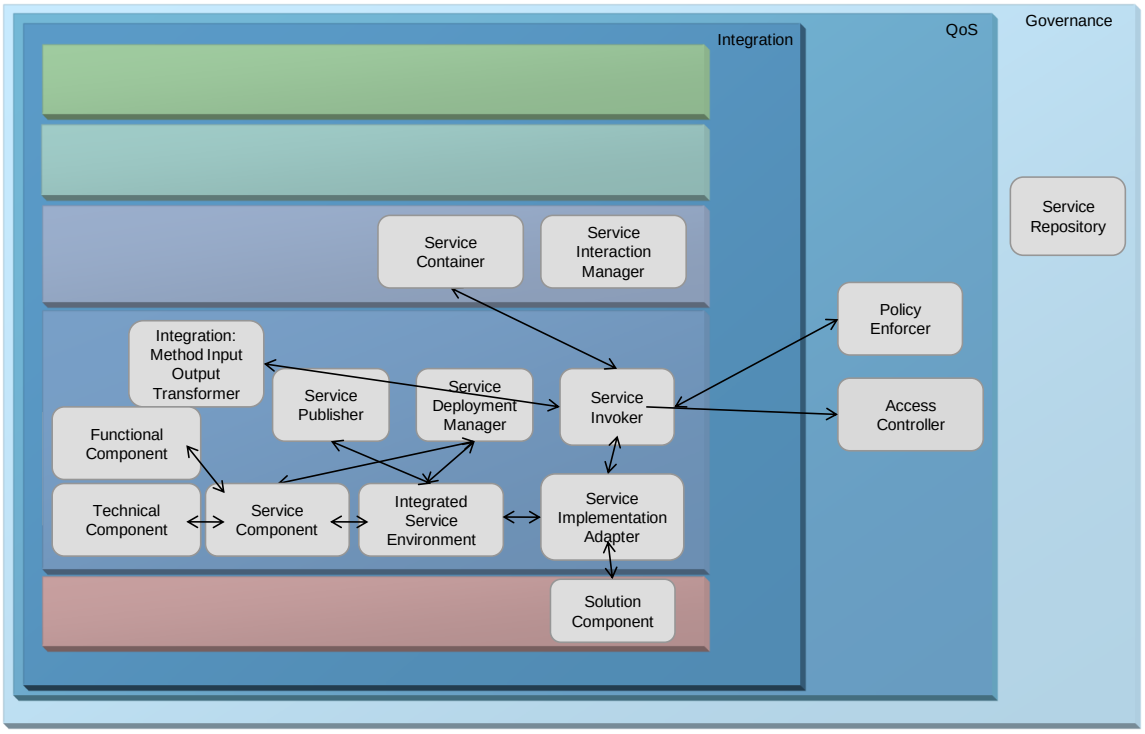


Figure 17: Key Interactions of the Service Component Layer with Horizontal Layers

9.4.2.1 *Interaction with the Services Layer*

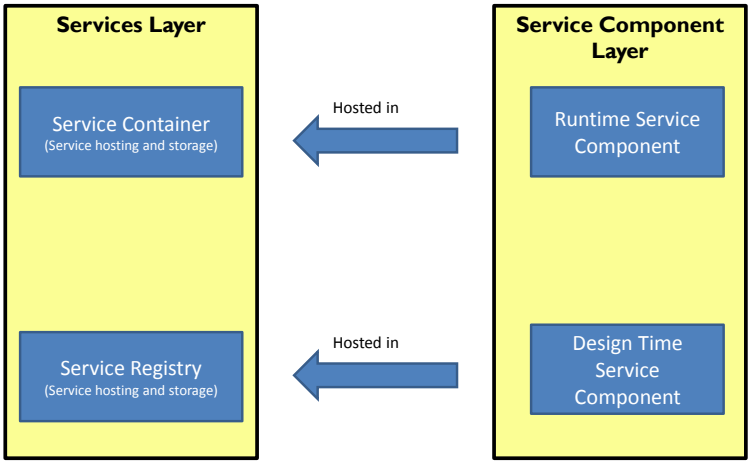


Figure 18: Relationships between the Services Layer and Service Component Layer

By its nature, this layer is coupled to the Services Layer of the SOA RA. A service definition change is likely to cause a direct side-effect on the Service Component in this layer. For example, if a service is retired from the Services Layer, the corresponding Service Component could also be retired if no other services are using it.[3] Finally, it is the responsibility of Service Components to faithfully reflect the definition of one or more services. To ensure that this relationship is maintained, a Service Component *must not* exhibit behaviors not defined in a service description.

The runtime relationship between the Service Component Layer and the Services Layer is illustrated in Figure 19. Services are deployed in the services container in the Services Layer. The services can be discovered using the Service Registry ABB in the Governance Layer; this can provide the contract and support for virtualization. The service then invokes the corresponding Service Component in the Services Component Layer which is then bound to the solution platform and invoked in the Operational Systems Layer.

[3] It can be argued that the service component may continue to provide values when the service has been retired, as it may continue to be used in the implementation of other service components or services. While this is true, it is important to recognize that the significance of the software component as a service-aligned component has been lost and as such it is no longer subject to the same governance as a service component.

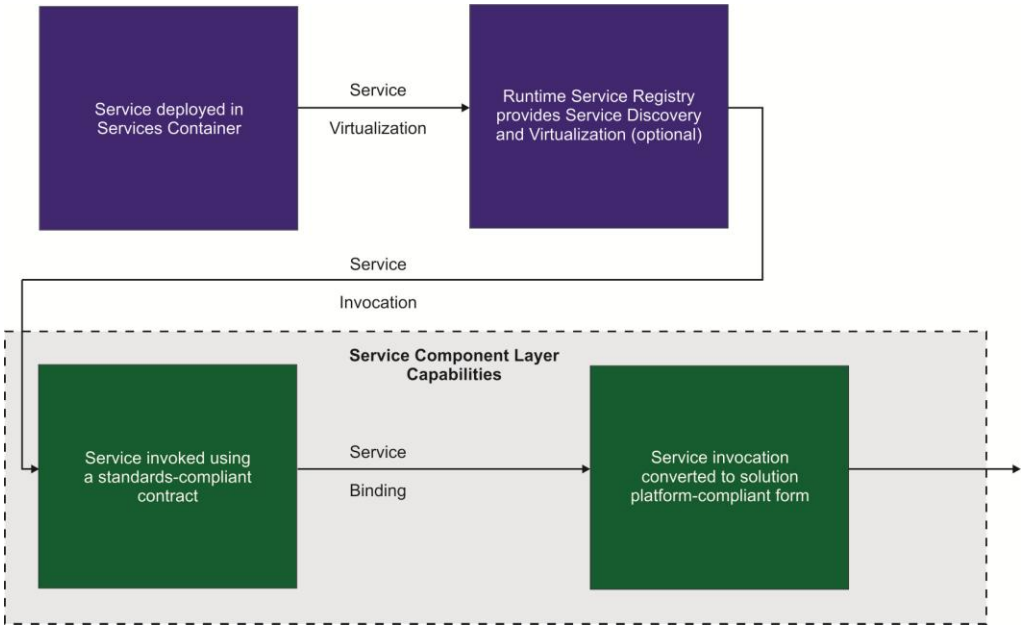


Figure 19: Use of Runtime Capabilities in the Service Component Layer

9.4.2.2 Interactions with the Operational Systems Layer

Service Components often consume behaviors from the Operational Systems Layer. This relationship creates a dependency on the behaviors consumed. If a decision is made to change the manner in which the Operational Systems Layer behavior is realized, there are likely side-effects on the Service Components that consume it. For this reason, traceability between the Service Component Layer and the Operational Systems Layer is an important element of an SOA.

Also, it is often the case that the Operational Systems Layer behavior required by a Service Component is not available in a convenient fashion. In such circumstances, the refactoring of the behaviors in the Operational Systems Layer may be necessary. This is an example of why the implementation of SOA may result in or require changes to the existing Operational Systems Layer.

9.5 Usage Implications and Guidance

9.5.1 Options and Design Decisions

There are multiple technology alternatives or realizations for the implementation of the Service Components. The selection criteria employed when choosing a realization technology would include a balance of the following criteria:

- **Capability**: The capability to realize the value proposition of Service Components and to realize the required behavior of a given service.
- **Familiarity**: Whether the capability of using the technology already exists in the organization.

- **Strategic**: Whether the technology is in line with the technology roadmap of the organization.
- **Manageability**: Whether the technology allows for the effective management of Service Components with respect to the Key Performance Indicators (KPIs) defined.

Often the selection requires the prioritization of these criteria. One alternative may offer features that are suited to the needs of a particular service component (e.g., message mediation) but not offer the management capabilities that other Service Components require (e.g., high availability).

Architectural decisions that are often made in connection with this layer include a choice of realization technologies, hosting, and runtime environments. As the Operational Systems Layer is connected with the four vertical layers which represent the cross-cutting concerns that are enablers for the functional layers, decisions regarding those cross-cutting layers will often involve the Operational Systems Layer. Therefore, this layer may be involved in questions pertaining to architectural decisions such as:

- What is the best hosting environment for a particular application?
- What is an appropriate runtime environment for a particular subset of an application?
- What kind of runtime capabilities are required in terms of NFRs?
- Should integration always occur through a Mediator ABB in the Integration Layer? Can it be performed inside a Service Component?
- Should collaborating Service Components consume each other through the Services Layer or through a platform-specific interface (e.g., EJB-to-EJB or EJB-to-Service)?
- Where are transformations performed? In the Service Component Layer or in the Integration Layer?
- Should component implementations be portable across multiple runtime environments?

Given that this layer alternately hosts the runtime for the implementation of the various services defined in the Services Layer, the salient KPIs are those that are of common concern in the enterprise systems, such as latency, availability, scalability, reliability, and security.

Latency refers to the delay of accessing a specific service due to internal implementation details and processes. Availability refers to the percentage of how a specific service can be available during a specific timeframe. Scalability refers to the ability of a specific service that supports various scales of consumer groups. Reliability refers to the ability of a specific service that stays no fail during a specific timeframe. Security refers to the ability of a service that provides an authorization and authentication facility to ensure secure access.

9.5.2 Implementation Considerations

9.5.2.1 Typical Interaction Sequences: A Narrative of the Flow

Refer to the example illustrated in Figure 20 where *Service A* is implemented using a combination of behavior from third-party *Package X* and *Application Y*. The consumer

Application B is coupled only to the description of the exposed service. The consumer must assume that the realization of the service is faithful to its published description and it is the providers' responsibility to ensure that such compliance is achieved. The details of the realization, however, are of no consequence to *Application B*. *Service Component A* acts as a service implementation façade; it aggregates available system behavior and gives the provider an enforcement point for service compliance. *Application B* invokes and interoperates with a service contract and specification defined in the interface in *Service A*.

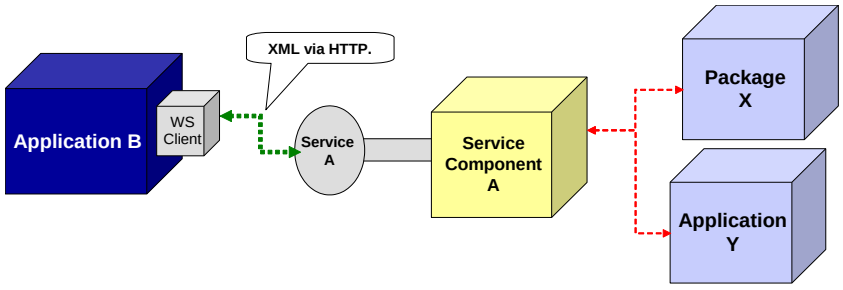


Figure 20: Service Components as a Facade

Subsequently, the provider organization may decide to replace "*Package X*" with "*Package M*". The required modifications are encapsulated in *Service Component A* with the result that there is no impact on any consumers of *Service A* such as *Application B*. This example illustrates the value of the Service Component Layer in supporting IT flexibility through encapsulation.

9.5.2.2 Composition Scenarios

Composition of existing application assets occur frequently in the context of transforming legacy systems into services. Figure 21 shows an example of how to construct Service Components using existing application assets. Assume two existing application systems – *Applicaiton 1* that maintains customer addresses, and *Applicaiton 2* that validates postal codes. These two existing applications were developed on proprietary platforms using proprietary technologies. In other words, these two applications are legacy systems. A new project intends to build two web services that can be accessed via the SOAP prototol. First is a "safe" address updating service that would validate postal codes before making changes to addresses. Second is to build a dedicated "validate postal code" service.

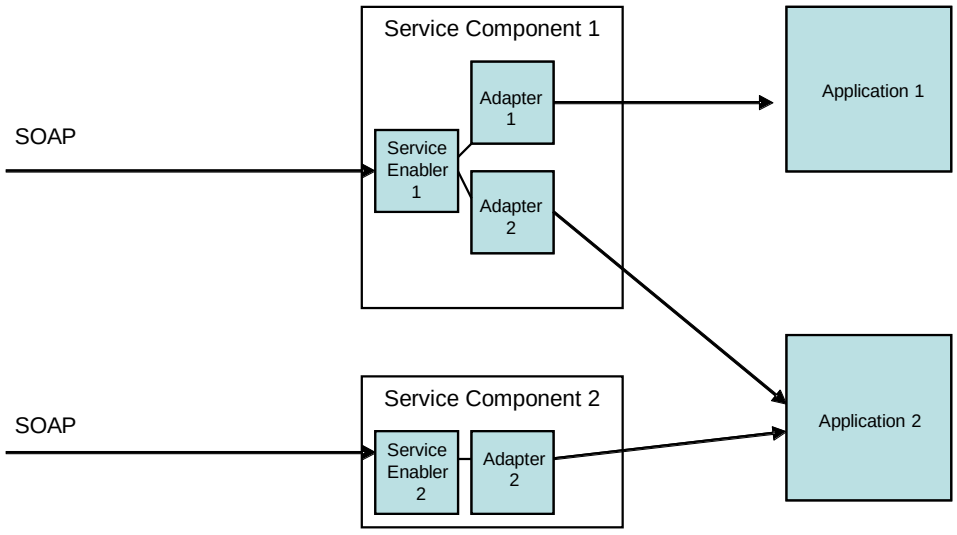


Figure 21: Interaction Flow in a Composition Scenario

As shown in Figure 21, there is a need to have an "adapter" component that represents particular existing application assets and provides an API for Service Components to consume and expose through necessary functions of this particular application assets. For example, both legacy applications have their own dedicated adaptors. "Adapter" is owned by the same organization that owns the application asset system and is legislated to be the only way to access this application asset. Such an adapter is provided as an API to the legacy system.

In order to construct a SOAP-enabled Service Component, a "service enabler" is a typical instrument, as shown in Figure 21. Each Service Component contains a service enabler component if it contains access to legacy systems through adapters. *Service Component 1* composes two legacy systems through dedicated adapters, and enables the composed service to SOAP access through a *Service Enabler 1*. *Service Component 2* composes two legacy systems through dedicated adapters, and enables the composed service to SOAP access through a *Service Enabler 2*. Note that *Application 2* only has one adapter, which is re-used in both *Service Component 1* and *Service Component 2*.

10 Services Layer

10.1 Overview

10.1.1 Context and Typical Flow

The Services Layer consists of all the services defined within the SOA. This layer can be thought of as containing the service descriptions for business capabilities and services as well as their IT manifestation during design time, as well as service contract and descriptions that will be used at runtime.

Service Components or existing enterprise applications (legacy systems, packaged applications, etc.) are responsible for the actual implementation *aka* realization of a service. At runtime, this implementation will reside in a container within the Operational Systems Layer, which is responsible for runtime.

The Services Layer is one of the horizontal layers which provide the business functionality supported in the SOA. The Services Layer is the layer of the SOA which describes functional capabilities of the services in the SOA. The Services Layer introduces the notion of services which are well-defined interfaces for a capability into the architecture with the advent of SOA. The notion of "programming to interfaces rather than implementation" only existed in the programming models such as Java and C++, but was never part of the architectural style until the advent of SOA and services.

This layer primarily provides support for services, from a design-time perspective. In particular, from a design-time perspective this includes assets including service descriptions, contracts, and policies. It defines runtime capabilities for service deployment, but the runtime instantiation of the Architecture Building Blocks (ABBs) enabling these capabilities are housed in the Operational Systems Layer. It also provides the service contract elements that can be created at design time to support subsequent runtime requirements.

Service specifications provide consumers with sufficient detail to locate and invoke the business functions exposed by a provider of the service. Ideally, this is done in a platform-independent manner. Service specifications may include:

- A description of the abstract functionality offered by the service similar to the abstract stage of a WSDL description [6]; note that the use of WSDL is illustrative and the description can be done in any language supporting description of the functionality
- A policy document
- SOA management descriptions
- Attachments that categorize or show service dependencies

Some of the services in the Services Layer may represent *versions* of other services in the set, which implies that a significant successor/predecessor relationship exists between them. The actual home for the various versions of a service should be sought in the Governance Layer which houses and centralizes the service registry and repository.

The Services Layer can be thought of as supporting categories of capabilities of the SOA RA:

- Functional capabilities *aka* services that enable business capabilities that business performs to achieve a business outcome or a milestone
- Supporting capabilities to define and specify the "services" in terms of service interface/contract/description, message specification, and policy descriptions
- Supporting capabilities to enable the runtime execution of the service and the support of service virtualization

These capabilities support the following main responsibilities of the Services Layer:

- To identify and define services
- To provide a container which houses the services
- To provide a registry that virtualizes runtime service access
- The provide a repository to house and maintain service design-time information

10.1.2 Capabilities

There are multiple categories of capabilities that the Services Layer needs to support in the SOA RA. These categories are capabilities which address the support of:

- **Service Definition**: This category of capabilities provides the ability to define the service description.
- **Service Runtime Enablement**: This category of capabilities provides the ability to support service versioning, to support service binding decoupling a service from its implementation, and provides the ability to provision services.
- **Policy Management**: This category of capabilities provides the ability to manage and enforce policies associated with services.
- **Access Control**: This category of capabilities provides the ability to manage access to services.
- **Service Clustering**: This category of capabilities provides the ability to cluster services.

This layer features the following supported capabilities:

Service Definition

1. Ability to define services in terms of service descriptions/contracts

Service Runtime Enablement

2. Ability to support the resolution of service versions so that, over time, as a service evolves, there is support for successive versions; this occurs when an existing service, with existing consumers, changes to the newly created version

3. Ability to enable the service container and the service registry to manage the storage and invocation of different services with minimal impact to users of the SOA

4. Ability to interact with other layers within the SOA RA, particularly the Integration Layer

5. Ability to define the binding to the Service Component that implements a given service

6. Ability to support the hosting of services

7. Ability to check the status and heartbeat of the services

Policy Management

8. Ability to support the integration of the Quality of Service (QoS) policy descriptions for services with the runtime elements of the Governance and the Quality of Service Layers

9. Ability to support standards that are compliant to consume the QoS policy descriptions and convert them into assets consumable by the ABBs within the layer

10. Ability to enforce the policies within the layer, behaving as a Policy Enforcer

11. Ability to support the audit and logging of runtime service usage to support QoS attributes, with the potential use of standards such as CBE and XDAS to ensure consistent and interoperable data which can then be easily integrated with the Quality of Service Layer to support capabilities such as service monitoring, audit, compliance, and runtime governance.

Access Control

12. Ability to support the integration of the security access control descriptions for services with the runtime elements of the Governance and Quality of Service Layers of the SOA RA

13. Ability to support standards that are compliant to consume the security policy descriptions and convert them into assets consumable by the associated ABBs within the layer

Service Clustering

14. Ability to cluster services which are contained by the service provider to invoke layers such as the Integration Layer; this capability enables the Services Layer to support the QoS requirements with regard to response and reliability

15. Ability to distribute services which are contained by the service provider to invoke layers such as the Integration Layer

10.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Services Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Service Definition	Service	1
2		Governance Layer: Service Repository	1
3	Service Runtime Enablement	Service Container	2-6
4		Service Interaction Manager	4
5		Governance Layer: Service Registry	3
6		Quality of Service Layer: Status Manager	7
7	Policy Management	Governance Layer: Policy Manager	8-9
8		Quality of Service Layer: Policy Enforcer	10-11
9	Access Control	Quality of Service Layer: Access Controller	12-13
10	Service Clustering	Cluster Manager	14-15

Table 3: ABB to Capability Mapping for the Services Layer

10.2 Details of ABBs and Supported Capabilities

10.2.1 Details of ABBs

This section describes each of the ABBs in the Services Layer in terms of their responsibilities.

10.2.1.1 Service

This ABB represents a published service that offers certain functionalities that business performs to achieve a business outcome or a milestone. This ABB is one of the core functional ABBs in SOA RA. Typically, a service is published to the Service Repository ABB in the Governance Layer during design time for search and re-use and the Service Registry ABB in the Governance Layer during runtime for service virtualizaton. A service is typically represented in a standard description language (e.g., WSDL) describing its accessible interfaces (e.g., function or method signatures). Service is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm.

10.2.1.2 Governance Layer: Service Repository

See Service Repository ABB in the Governance Layer.

10.2.1.3 Service Container

This ABB acts as a container or gateway by providing the environment with the ability to invoke and run services (manage their runtime invocation, lifecycle, etc.). The Service Container is also commonly known as a Service Gateway. The primary responsibility of the Service Container is to encapsulate the code that implements the low-level details of communicating with a service into this ABB. Some Service Containers require capabilities beyond basic communication, such as transactions and security.

Thus, its key communication and virtualization responsibilities include the invocation and execution of services, encapsulating the components that implement the service (i.e., providing the service end-points), state management, and the binding of service invocations to cross-cutting layers (such as the Integration Layer and the Business Process Layer in particular), the clustering of services, and their distribution to different consumers.

It should be noted that all ABBs (including the Service Container) are instantiated in the Operational Systems Layer. For instance, a Service Container may be contained within a J2EE environment or a .NET environment. It is also possible for it to be a hardware device as long as it provides the ABBs required with the ability to support runtime invocation and running of services and integration with cross-cutting layers.

Within the Service Container are ABBs which enable it to invoke and execute service components, and support the integration with the cross-cutting layers – the Quality of Service Layer, Integration Layer, and Governance Layer. It leverages the Service Repository ABB and Service Registry ABB in the Governance Layer to support service versioning and virtualization.

10.2.1.4 Service Interaction Manager

This ABB is contained within the Service Container and in general manages the interactions required to invoke and run the services. It uses all the other ABBs within the Services Layer to achieve its goals.

10.2.1.5 Governance Layer: Service Registry

See Service Registry ABB in the Governance Layer.

10.2.1.6 Quality of Service Layer: Status Manager

See Status Manager ABB in the Quality of Service Layer.

10.2.1.7 Governance Layer: Policy Manager

See Policy Manager ABB in the Governance Layer.

10.2.1.8 Quality of Service Layer: Policy Enforcer

See Policy Enforcer ABB in the Quality of Service Layer.

10.2.1.9 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

10.2.1.10 Cluster Manager

This ABB supports scalability within the Services Layer. It provides clustering and caching support when necessary.

10.2.2 Structural Overview of the Layer

The ABBs in the Services Layer can be thought of as being logically partitioned into categories which support the abilities to identify and specify services during design time and to provide a runtime environment for services and abilities to managing service metadata in support of the service runtime.

The service runtime environment needs to:

- Provide runtime support for services
- Provide the container to support runtime service lifecycle management
- Separate out service types and versions and invoke these services
- Support scalability, which becomes critical with high-volume service invocations
- Integrate cross-cutting concerns, allowing access control, audit and identity integration (security policies), and QoS policies to be integrated
- Support the actual conversion and binding to the platform for an individual service

Thus, the ABBs in the Services Layer enable design-time capabilities, such as service definition, and runtime capabilities, such as service container, providing a runtime environment for services. The Service ABB along with the Service Repository ABB in the Governance Layer supports design-time capabilities and the Service Container ABB along with the Service Registry ABB in the Governance Layer support runtime capabilities.

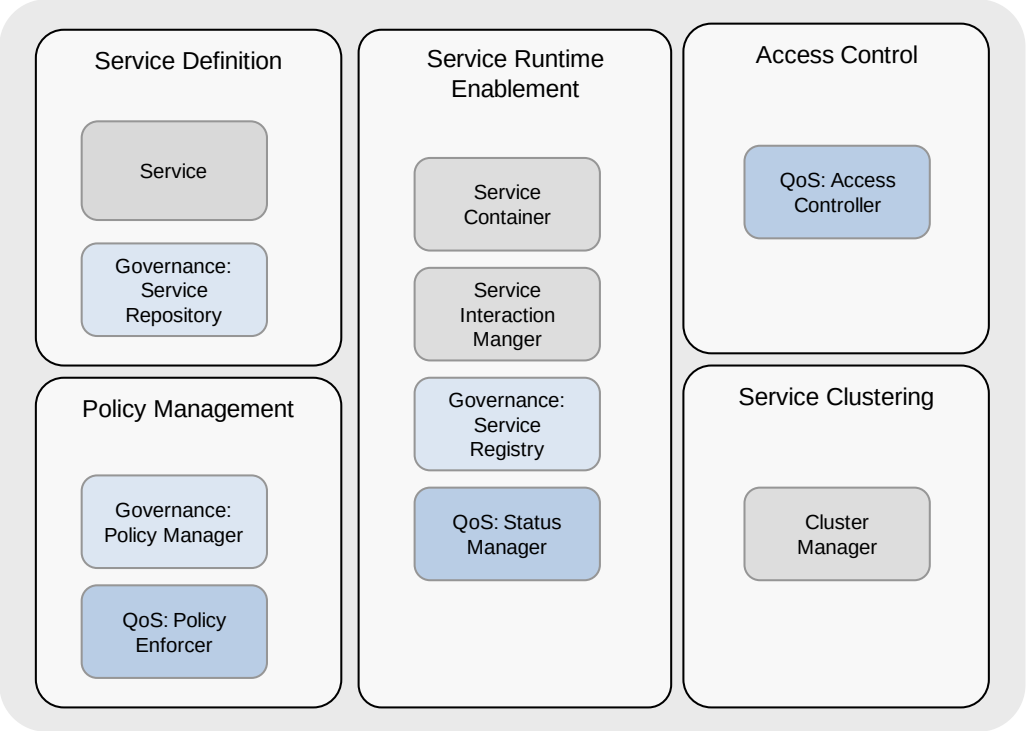


Figure 22: ABBs in the Services Layer

Figure 22 illustrates the ABBs in the Services Layer and ABBs from other layers that are core to fulfilling the responsibilities of the Services Layer.

- ABBs supporting design-time needs are:
  - — Service ABB
  - — Service Repository ABB in the Governance Layer
  - — Policy Manager ABB in the Governance Layer
- ABBs supporting runtime environment for the services are:
  - — Service Container ABB
  - — Service Interaction Manager ABB
  - — Service Registry ABB in the Governance Layer
  - — Policy Enforcer ABB in the Governance Layer
  - — Access Controller ABB in the Quality of Service Layer
  - — Cluster Manager ABB
  - — Status Manager ABB in the Quality of Service Layer

10.3 Inter-Relationships between the ABBs

Figure 23 shows the inter-dependencies between the ABBs during design time and runtime.

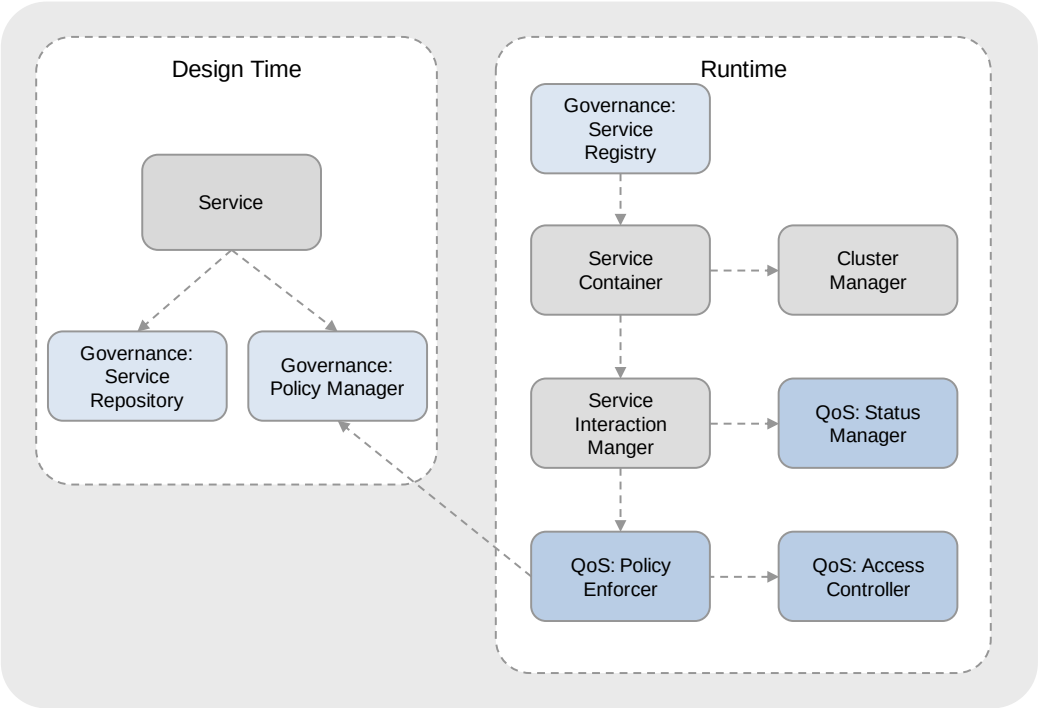


Figure 23: Relationships among ABBs in the Services Layer

During design time, information such as metadata about service contracts gets stored in the Service Repository ABB in the Governance Layer and policy associated with services are defined using the Policy Manager ABB in the Governance Layer.

During runtime, the consumers of services interact with the Service Registry ABB in the Governance Layer to find the service. The Service Registry ABB then invokes the service hosted in the Service Container ABB where the Service Interaction Manager ABB then manages the interaction between the various ABBs within the container and other layers of the SOA RA. The Policy Enforcer ABB in the Quality of Service Layer enforces service policies (including both QoS and security policies). The Service Container ABB invokes the Service Component in the Service Component Layer to realize a service. Thus the functionality of the service and the physical service is the Service Component, while the role of the Services Layer is to act as the translation between the consumer and the Service Component. Upon completion of execution the service is propagated back up to the Service Binder, and then the Service Interaction Manager which applies policies using the Access Manager and the Policy Enforcement End-point, logs compliance and runtime logging information using the Policy Manager, and finally propagates the information via the Service Binder back to the Service Consumer.

Usage of a service by a consumer involves two steps – service discovery and location, and service invocation. The figures below show these steps.

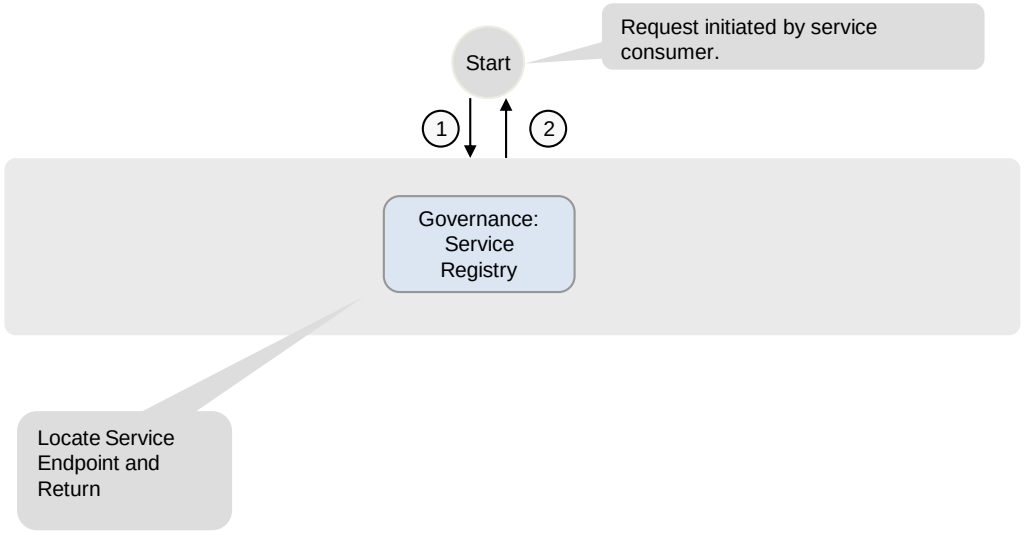


Figure 24: Interaction Flow for Service Discovery and Location

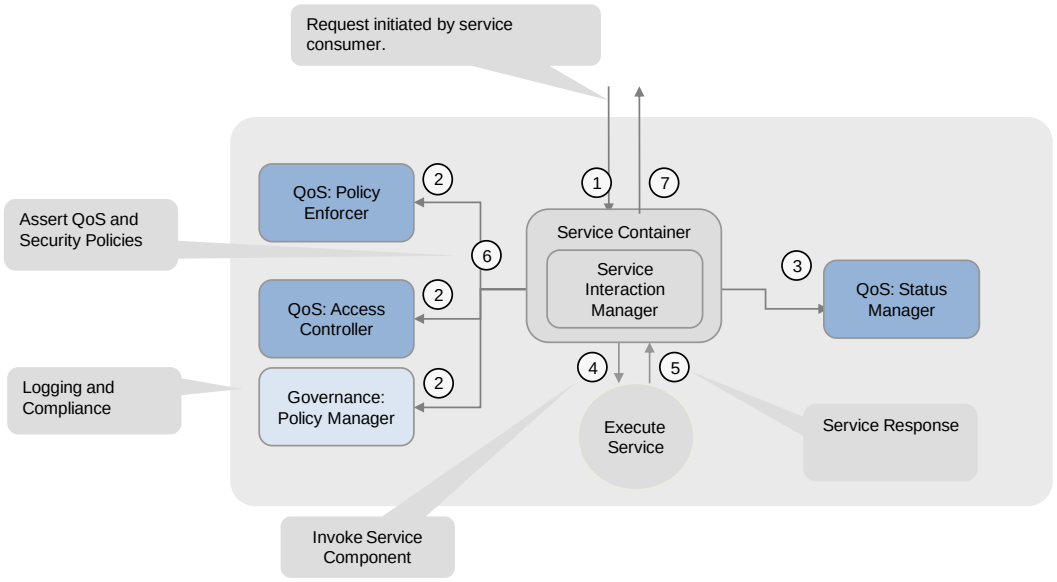


Figure 25: Interaction Flow for Service Invocation

10.4 Significant Intersection Points with other Layers

10.4.1 Interaction with Cross-Cutting Layers

The Service Repository ABB in the Governance Layer acts as the interaction point at design time with the Information, Governance, and Quality of Service Layers, respectively. The Service Container interacts with the Integration Layer using ABBs such as the Service Integration Controller. The Service Container ABB uses the Service Repository and Registry in the Governance Layer to find information needed to support the service, such as policies and binding information. This relationship at runtime enables late binding of services.

The Policy Manager ABB in the Governance Layer, Access Controller ABB in the Quality of Service Layer, and Policy Enforcer ABB in the Quality of Service Layer exchange and enforce policies ensuring standards-compliant interaction that adheres to the governance regimen. Figure 26 illustrates these relationships.

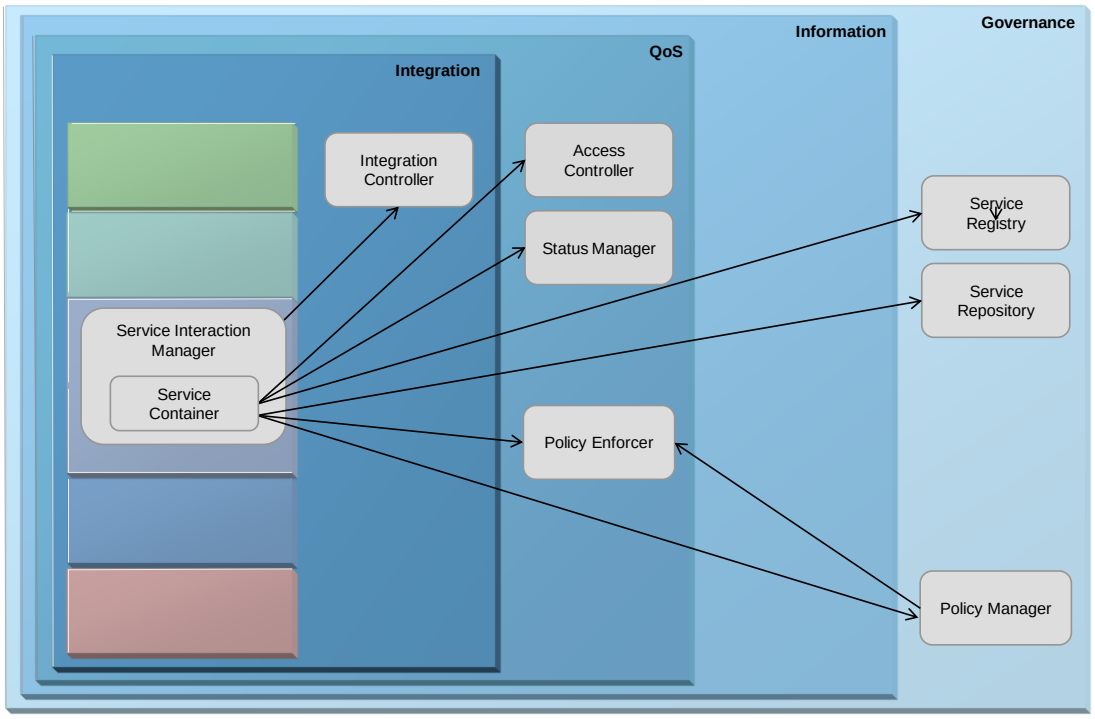


Figure 26: Interactions from the Services Layer to the Cross-Cutting Layers

The Service Container ABB also leverages the Policy Enforcer ABB to enforce the service policies in order to deal with compliance of Service-Level Agreements (SLAs) of services.

10.4.2 Interaction with Horizontal Layers

The Service Registry ABB in the Governance Layer is where service consumers interact with the Services Layer to find the service end-point, by which a service is invoked (the actual specification of what is a service end-point varies based on the actual solution architecture and the resultant solution platforms). The Service Interaction Manager is the invocation integration point for the Service Container ABB which then manages using the Service Interaction Manager to coordinate all its internal ABBs. The Service Interaction Manager invokes the Access Controller ABB and the Policy Enforcer ABB in the Quality of Service Layer to assert the QoS contract of the service and to invoke the Service Interaction Manager to convert invocations into Service Component invocations, thus effectively executing the associated service functionality. Finally, Service Components upon completion of the service execution send back the service response data to the Service Interaction Manager which then propagates the service response data back to the service consumer. During execution, if the status of the service changes, the Service Interaction Manager notifies the Status Manager in the Quality of Service Layer of the change. Likewise, the Status Manager can interact with the Service Interaction Manager to change the status of the service.

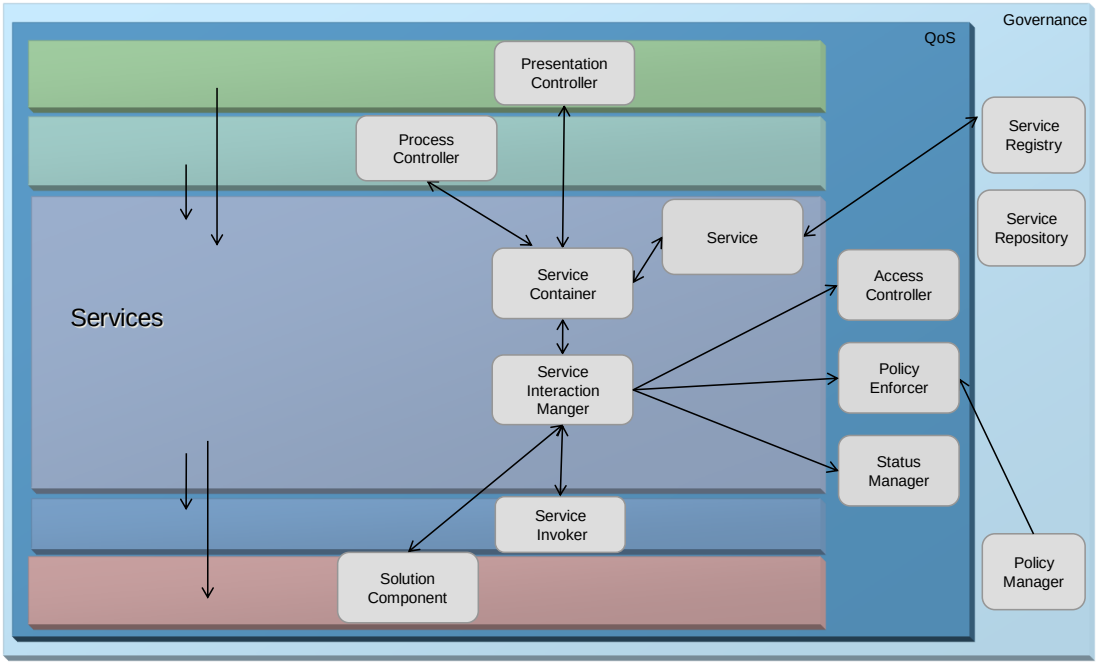


Figure 27: Interaction with Horizontal Layers

To summarize:

The Service Repository ABB in the Governance Layer provides design-time storage of metadata for services.

The Service Registry ABB in the Governance Layer supports the storage of and access to bindings at runtime to services hosted in the Service Container/Gateway ABB. It manages service versioning allowing the appropriate service to be picked.

The Service Interaction Manager ABB plays three roles – acting as the outward-facing ABB which is invoked by other layers at runtime to invoke services, and invoking the service components from the service components layer, and converting data between different formats. The service binder invokes the service container to compose all the information required to invoke the service ABB. The Service Interaction Manager ABB uses the Policy Enforcer ABB and Access Controller ABB in the Quality of Service Layer to enforce and incorporate any security and QoS policies. It uses the state manager to address any state-related issues. It then calls the Service Invoker ABB in the Service Component Layer to execute the service functionality, accept the result from the service component, interact with the Service Interaction Manager ABB, and propagate it back via the Service Container ABB to the invoking consumer.

The runtime services are housed in a Service Container ABB using the hosting ABB. The Service Container ABB manages the runtime service lifecycle and uses the Service Interaction Manager ABB to invoke Service Components and the cluster manager to support scalability in the service container.

The Policy Enforcer ABB provides the interaction point, between the Quality of Service Layer and the Service Layer, supporting Policy Enforcement. The Access Manager provides Authorization and Authentication support in the context of the layer and integrates with

corresponding ABBs which define security policies in the Quality of Service Layer. In practice, it too acts as a Policy Enforcer.

10.5 Types of Service

As outlined in Section 10.2.2, one type of ABB in the Service Layer is a service. Services are naturally a key concept in any SOA and it is important to realize that there can be many different kinds. This section defines a standard categorization scheme for services. Services are categorized according to what they do; i.e., their function or purpose, in order to aid in ensuring both coverage and shared understanding. Of course, other categorization schemes are also possible and helpful.

Partitioning services into groups is a common activity in the development of the services and service portfolio in an SOA. Categories and groups of services affect how both business and IT views and understands the architecture and the portfolio of services that supports it.

Note that the categorization scheme for a particular type of ABB is distinct and separate from which layers in the reference architecture such ABBs are associated with. For example, there is no contradiction in defining "process services" as a category of services. A process service is simply process logic exposed as a service. Since these are in fact services, as service ABBs they belong in the Services Layer. Yet, having said that, their realization as processes properly belongs in the process layers and typically leverages other ABBs such as a Process Controller. In fact the process service implementations will often also rely on implementations of ABBs in other layers, like the Process Manager and policy ABBs in the Quality of Service Layer. The seeming dichotomy is not a dichotomy at all, but simply a natural consequence of delineating the service itself (as a service) from the realization of that service (as a process). This is consistent with The Open Group SOA Ontology [24] where there is a clear delineation between the logical service itself and things that perform that logical service.

Figure 28 shows a functional categorization scheme for services found in a typical enterprise. As mentioned above, this categorization scheme is for the services themselves, not for their implementations (which will be ABBs of other layers of the SOA RA).

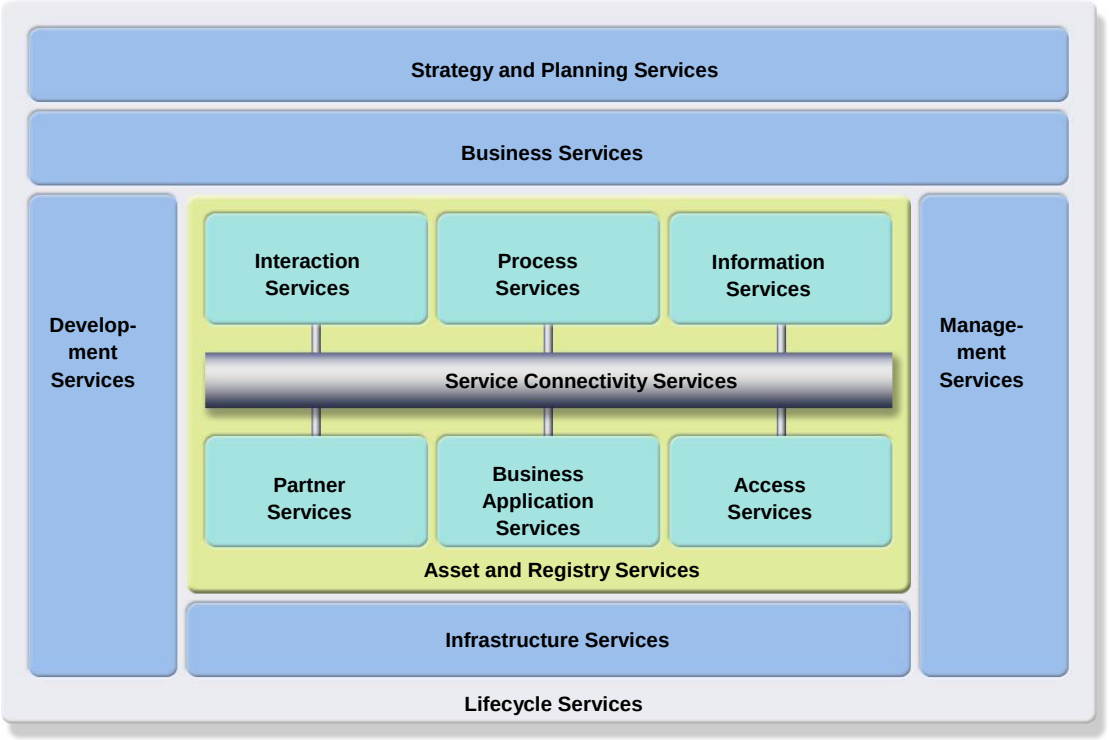


Figure 28: Functional Categorization Scheme

The categorizations of services are broken down in Figure 28. Those categories (such as the interaction services, process services, etc.) graphically connected to the Service Connectivity Services and Business Services category are considered to be domain-specific. Services in the domain-specific categories are solution-specific and thus require unique implementation-specific ABBs to implement their semantics.

The remaining services categories are considered to be domain-neutral. These domain-neutral categories include development services, management services, etc. Services in these categories can be used in any domain or solution. In general, domain-neutral services are used to plan, develop, support, and manage the domain-specific services in the solution.

Note that the Interaction, Process, and Information service categories support the Model-View-Controller Pattern. The value of separating these aspects in the traditional view of architecture still holds true for SOA.

Services categories are described below.

10.5.1 Interaction Services

Interaction services are a category of services that provide the presentation logic of the business design. These services are components that support the interaction between applications and end users. Interactions with the external world are not limited to just interactions with humans; interaction logic orchestrates the interface to all kinds of devices and control systems, including vehicles, sensors, and RFID devices. Every external interaction projects a view of the

information system that is tailored for the specific interaction fidelity, frequency of interaction, and presentation composition that best fits the needs of the end user or device.

Interaction services may also be tailored to the situation as well as role-sensitive contexts. Adjusting what is seen and the behavior presented to the external world based on who the user is, what role they are performing, and their location. Authentication, privilege selection, and proximity may all be significant to what users can do and how. Collaboration and collaboration services also can be categorized as interaction services as they also provide a means for users to interact with the solution.

Interaction services are most closely aligned with the Consumer Layer. Interaction service implementations use the Presentation Controller ABB in the Consumer Layer to present the interface. The Presentation Controller ABB uses ABBs from other layers to complete its implementation; for example, it uses the Access Controller and the Policy Enforcer ABB implementations from the Quality of Service Layer to provide the support for role-sensitive content and authentication. Interaction service implementations may also use the Integration Controller and Mediator ABB implementations from the Integration Layer to communicate with the consumer. Together, these ABBs, and others, work across the layers to allow a user to interact with a given system.

10.5.2 Process Services

Process services are a category of services that include various forms of compositional logic. The most notable of which are business process flows, business state machines, business rules, and decision tree processing. It is appropriate to select the abstraction that best matches the implementation of the business design.

Process services and their composition abstraction preferences and the business logic where business rules are enforced have a tight integration with the business. The rate of change, administration requirements, and legal control of the logic behind these rules dictates whether another paradigm should be used to create and govern these rules.

Business rule engines are one way to customize a business process abstraction; for example, a business check, such as *isItemTaxable*(), can be inserted in the business logic, and rely on the business rules engine to consult a separately managed table of tax rules, which will return whether or not sales tax should be applied to the purchase. This table is managed by a business administrator who has the proper business authority rather than a business logic programmer – thus, separating the concerns of the business logic from the rules that govern the logic. This enables dynamic processes and support for decision services to make or advise on decisions in processes or at the end of processes.

Process services are most closely aligned with the process layer. Process service implementations implement or use implementations of the Process Controller and Process Flow Manager ABBs. In turn, these ABBs in the process layer implement ABBs from other layers, like the Business Rules Manager, where business rules are defined, in the Governance Layer.

10.5.3 Information Services

Information services are a category of services that contain the data logic of business design. The service implementations that provide the data logic have three major responsibilities: to provide

access to the persistent data of the business, to support data composition of the business, and to provide their own sub-architecture for managing the flow of data across the organization.

- **Data Access**: The data access information service implementations can include query statements for retrieving information or referential integrity checks on the information manipulated by these service implementations. Information services for data access incorporate federation of multiple data sources.
- **Data Composition**: The data composition information service implementations compose information in a way that matches the composition of services in the business design. This is analogous to the kind of re-factoring that can occur with legacy applications to get them to fit better with the business design. In addition, it is common practice to implement these services to separate the database design from the application design to achieve the level of performance and scalability required in many enterprise computing environments.
- **Data Flow**: The data flow information service implementations manage the movement of information from one part of the enterprise to another. The movement of data is needed to satisfy its own data flow and lifecycle requirements. This may involve the use of Extract-Transform-Load (ETL) mechanisms to process and enrich data in bulk, batch processing activities involved in bulk transaction processing, and migrating data from master-data-of-record databases to information warehouses that can be used to perform post-processing and business intelligence, analytics, and content management functions – which in turn are made available to the business application as services.

Information services are most closely aligned with the Information Layer. Information service implementations use the ABBs in the Information Layer. These ABBs include but are not limited to the Data Validator, Data Aggregator, Content Manager, Data Repository, and Data Federation. Together these ABBs provide the means for the service implementations to find and present data in a logical manner.

10.5.4 Business Application Services

Business application services are a category of services that implement core business logic. These are service implementations created specifically within a business model and that represent the basic building blocks of business design. These services are not decomposable within the business model, but can be composed to form higher-level services. Often implementations of these services will be composed in business processes (such as process flows or business state machines). However, these service implementations may also be invoked directly by presentation logic in interaction services.

Business application services are most closely aligned with the Services Layer. Business application service implementations implement or use implementations of the Service Container and Service Interaction Manager ABBs. Business application service implementations may also implement or use implementations of ABBs from other layers, including the Access Controller and Policy Enforcer from the Quality of Service Layer as well as the Policy Manager from the Governance Layer.

10.5.5 Access Services

Access services are a category of services that are dedicated to integrating legacy applications and functions into the SOA solution. This can be as simple as wrapping those functions and rendering them as service implementations. This can also be a more complex case that augments the logic of the existing function to better meet the needs of the business design. In other architectures we have often referred to these access service implementations as adapters. In the SOA RA, these service implementations are distinctly responsible for rendering these adapters so that they can be manipulated and composed within business processes like any other service implementation component.

Access services are most closely aligned with the Services Layer. Access service implementations implement or use the implementation of the Service Invoker ABB in the Component Layer which interacts with other ABB implementations within the Component Layer for binding to the service interface and accessing the Operational Systems Layer's Solution Component and Hardware ABBs.

10.5.6 Partner Services

Partner services are a category of services that capture the semantics of partner interoperability that have a direct representation in the business design. These services include the policies and constraints that other businesses must conform with to work within the business. Partner service implementations are somewhat analogous to interaction service implementations in that they project a view of the business to the partners, and control the interaction with them as an external entity. Partner service implementations are also analogous to access service implementations because they render the capabilities of that partner as a service so that those functions can be composed into the business processes.

Partner services are most closely aligned with the Services Layer. Partner service implementations use the Service Interaction and Service Container ABBs. These ABB implementations also use or implement ABBs from other layers, including the Integration Controller in the Integration Layer, the Access Controller and Policy Enforcer from the Quality of Service Layer, and the Service Registry and Repository and Policy Manager from the Governance Layer.

10.5.7 Service Connectivity Services

Service connectivity services are a category of service that assumes the responsibility for binding service consumers with service providers – they implement this responsibility by resolving their location automatically to achieve an optimal routing of requests across the network and meet the goals of the business. The presence of service connectivity services, like Enterprise Service Buses (ESBs), should be transparent to the consumer of functional services in the SOA solution. Though transparent, service connectivity services are fundamental to simplifying the task of invoking services – making the use of services wherever they are needed.

Implementations of the service connectivity services support interconnectivity and host Mediations – logic that may perform message transformation, intelligent routing, augmented functionality (such as logging or auditing) to enable the interconnectivity of services. Because the service connectivity services are a transparent fabric interconnecting service consumers with service realizations/implementations, then, by extension, the service connectivity service

implementation also makes hosting mediating logic, and more specifically the hosting topology, transparent to the service consumers and providers which are being mediated. Service connectivity services are a mix of domain-neutral (messaging products and ESBs available from many vendors), and domain-specific (the implementations of adapters needed from existing services and operational systems into the ESB).

Service connectivity services are implemented mostly with the ABBs in the Integration Layer. The ABBs used could include the Integration Controller, Mediator, Router, Adapter, Data Aggregator, and Message Transformer, depending on the exact needs.

10.5.8 Asset and Registry Services

Asset and registry services are a category of services that provide access to the assets that are part of the overall architecture. Implementations of these services provide access to service descriptions, software services, policy, documentation, and other assets or artifacts that are essential to the operation of the business. These are assets and artifacts that need to be registered for search and consumption and therefore need to be managed (usually by services in the lifecycle category). Services that provide access to these assets are especially important in a heterogeneous environment as they allow for the query of assets within the environment across multiple registries. Once located, these assets can then be incorporated into the overall SOA and invoked to provide necessary function for the business. It is important to note that asset and registry services are used by lifecycle service implementations, but they do not provide lifecycle services themselves.

Asset and registry services implementations implement or use implementations of the ABBs from the Governance Layer such as the Service Repository ABB and the Service Registry ABB.

10.5.9 Infrastructure Services

Infrastructure services are a category of services that form the core of the information technology environment for hosting SOA applications. It is through these service implementations that a reliable system can be built to provide efficient utilization of resources, ensure the integrity of the operational environment, and balance workload to meet service-level objectives, isolate work to avoid interference, perform maintenance, secure access to confidential business processes and data, and simplify overall administration of the system.

Infrastructure services virtualize the underlying computing platform and resource dependencies. The service implementations themselves are built using SOA principles – exploiting the characteristics of loose coupling to enable highly flexible and composable systems.

The SOA RA has been designed to specifically allow different technologies to be plugged at various layers of the system – allowing the trade-off of tight-integration QoS with the flexibility to pick-and-choose which mix of product technologies are appropriate for the business requirements and goals, and to address the inevitable heterogeneity of legacy environments.

Infrastructure services are most closely aligned with the Operational Systems Layer. Infrastructure services implement or use implementations of the Solution Component, Implementation Controller, as well as Hardware and Virtualized Infrastructure ABBs. Infrastructure service implementations implement or use implementations of many of the ABBs in the Quality of Service Layer to provide the management of the infrastructure services and

underlying resources; i.e., IT systems manager, availability manager, and performance manager. These ABBs work together to provide an overall IT environment for hosting an SOA solution.

10.5.10 Management Services

Management services are a category of services that represent the set of management tools used to monitor service flows, the health of the underlying system, the utilization of resources, the identification of outages and bottlenecks, the attainment of service goals, the enforcement of administrative policies, and recovery from failures. This includes in a business process management context the management of business processes and monitoring of performance metrics and Key Performance Indicators (KPIs). Management service implementations can be used to help prioritize the resolution of problems that surface in the information system, or to direct the allocation of execution capacity to different parts of the system based on service-level goals that have been set against the business design.

Management services are most closely aligned with the Quality of Service Layer. Management services implementations implement or use implementations of some of the ABBs in the Quality of Service Layer, including the Command and Control Manager, IT Systems Manager, Event Manager, Policy Enforcer, Configuration Manager, Security Manager, and Solution Manager, which include an Availability Manager, Reliability Manager, and Performance Manager. These ABB implementations then rely on the Service Repository and Policy Manager ABB implementations in the Governance Layer to help implement the management services.

10.5.11 Business Services

Business services are a category of services that capture the business function and then introspect on the business design to improve it through a combination of iterative refinement and analysis of real-time business metrics.

Management service implementations in the Quality of Service Layer help to define KPIs; that is, those business objectives and general metrics that are desired to be monitored. The service implementations will be linked directly into the information system to both collect performance metrics coming out of the system as well as to enable the change of which metrics are measured as monitoring needs change. Automated analysis of these metrics could automatically suggest improvements to the business design to better meet the business goals and objectives. However, capturing them for consideration by business executives, business analysts, and other human experts obviously meets an immediate and long-standing need, and is an incremental step toward the automation and flexibility promised by SOA.

Business services are consumers of the functional services outlined in the previous section and closely aligned with the Consumer Layer for implementation ABBs. Business service implementations also use the management ABBs in the Quality of Service Layer to implement the support for metrics and metrics monitoring; i.e., the Monitoring Metrics Tools ABB, the Policy Enforcer ABB, and the Business Activity Manager ABB. These ABBs work across the layers to provide and monitor business services.

10.5.12 Strategy and Planning Services

Strategy and planning services are a category of services that supports creating a vision, blueprint, and transition plan for improving business outcomes. Specifically, these are services

that process the strategies of the business to create an implementation roadmap covering both business and IT. In other words, these services support the long-term evolution and effectiveness of an enterprise.

Strategy and planning services produce strategies and enterprise blueprints that define a desired future state and are used to prioritize, select, guide, and govern the execution of projects – the purpose is the planning of effective change. Examples of enterprise blueprints are work products such as component business models, business architectures, and enterprise architectures, all created with the purpose of achieving business and IT alignment and better business outcomes.

Strategy and planning services are typically used (or produced) by roles such as strategists, enterprise architects, and business architects. Included in the category of strategy and planning are services for governance, architectural, and organizational change. Included in this category are also services that support collaboration and coordination across planning and delivery.

Strategy and planning services are most closely aligned with the Governance Layer and allow business and IT to plan and prioritize changes to solutions and operations. The Policy Manager and Business Rules Manager, Reporting Tools, and Change Control Manager ABBs are used to implement and provide these strategy and planning services.

10.5.13 Development Services

Development services are a category of services that encompass the entire suite of architecture tools, modeling tools, development tools, visual composition tools, assembly tools, methodologies, debugging aids, instrumentation tools, asset repositories, discovery agents, and publishing mechanisms needed to construct an SOA-based application. Some development tools, like Eclipse, have a built-in mechanism for modularizing and plugging in tool services, thus encouraging the construction of the development tools as services following many of the same principles promoted by SOA.

Development services use repository ABBs in the Governance Layer to get the descriptions needed during development. Development service implementations can also register the appropriate services in the appropriate service registries by leveraging asset and registry services, or possibly even directly via the Service Registry ABB.

10.5.14 Lifecycle Services

Lifecycle services are a category of services that support managing the lifecycle of SOA solutions and all of the elements that comprise them across development and management, ranging from strategy to infrastructure. Lifecycle services can be applied to all categories of services, managing and governing the service definitions and service implementations within that category. Managing and governing the full lifecycle of an SOA solution includes SOA Governance, Policy Management, Requirements Management, and Configuration Management.

Implementations of the lifecycle services rely strongly on asset and registry service implementations since these provide access to some of the portfolio of assets that the lifecycle service implementations must manage. The assets that are managed include service implementations, processes, documents, etc.

Lifecycle services are most closely aligned with the Governance Layer. The Service Repository and Service Registry ABBs are used to implement and provide lifecycle services.

10.5.15 Summary

These categories or types of services can be kept in mind when developing your service portfolio and your SOA solution portfolio. Use these as a checklist to ensure that you have considered all the possible services and can make the right choices on fulfilling the development or purchase of those services. The examples of which SOA RA layers and ABBs can be used to develop the services should make selecting the right ABBs easier for your solution architecture.

10.6 Usage Implications and Guidance

Exposed services reside in this layer. They can be discovered and invoked, or possibly choreographed to create a composite service. Services are functions that are accessible across a network via well-defined interfaces of the Services Layer. The Services Layer also provides for the mechanism to take enterprise-scale components, business unit-specific components, and in some cases project-specific components, and externalizes a subset of their interfaces in the form of service descriptions. Thus, the components provide services through their interfaces. The interfaces get exported as service descriptions in this layer, where services exist in isolation (atomic) or as composite services.

For example, a service container, in which the services are hosted and invoked from, is also a part of the Services Layer. The service container is compliant with the standards for service specification being supported by the service, and runs on a hosting platform in the Operational Systems Layer.

This layer contains the contracts that bind the provider and consumer. Services are offered by service providers and are consumed by service consumers (service requestors).

The layers and their underlying building blocks in the target architecture may be defined according to the service identification activities which may be defined through three complementary techniques of domain decomposition, existing asset analysis, and goal-service modeling to identify, specify, and realize services, components, and flows [1][20]. They represent the heart of the SOA value proposition, which is improved agility via the decoupling of business and IT. The quality of these service definitions will have a significant impact on the benefit of the given SOA effort.

Services are accessible independent of the implementation and transport. This allows a service to be exposed consistently across multiple customer-facing channels such as the web, Interactive Voice Response (IVR), etc. The transformation of response to HTML (for the web), VoiceXML [16] (for IVR), and XML string (for XML client) can be done via XSLT [19] functionality supported through transformation capability in the Integration Layer.

It is important to acknowledge that Service Components may consume services to support integration. The identification and exposure of this type of service; i.e., the internal services, does not necessarily require the same rigor as is required for a business service. While there may be a compelling IT-related reason behind the use of such services, they are not generally tied

back to a business process and as such do not warrant the rigorous analysis required for business services.

In addition to being an important template for defining an SOA solution at a logical level, the SOA RA is also a useful tool in the design of vendor-neutral SOA solutions. This is because it enables the objective identification of SOA infrastructure requirements. The SOA RA provides a well-factored decomposition of the SOA problem space, which allows architects to focus on those parts of an SOA solution that are important in the context of the problem they are solving and to map the required capabilities onto vendor product capability – rather than try and reverse engineer an SOA solution architecture from the capability of a particular vendor's products. This set of requirements can be used to better leverage the various capabilities provided by a mix of different vendors who may offer the same ABB. Using the same SOA RA, SOA business services can be delivered based on the same deployment framework.

11 Business Process Layer

11.1 Overview

11.1.1 Context and Typical Flow

With the advent of SOA came the promise of agility and flexibility. An organization that has embarked on the journey of SOA would be successful in delivering the promise of agility and flexibility only when its business processes and associated flows are realized in the architecture in a fashion that allow rapid changes in the implementation of these business processes. In the past, business process flows were typically embedded in software components and user interfaces. The SOA Reference Architecture (SOA RA) recognizes the promise of SOA and the challenges in the past in the ability to change the business processes as market changes and therefore introduces a specific layer for business processes and flows. The layer is called the Business Process Layer and allows externalization of the business process flow in a separate layer in the architecture and thus has a better chance to rapidly change as the market condition changes.

The Business Process Layer covers process representation and composition, and provides building blocks for aggregating loosely-coupled services as a sequencing process aligned with business goals. Data flow and control flow are used to enable interactions between services and business processes. The interaction may exist within an enterprise or across multiple enterprises.

This layer includes information exchange flow between participants (individual users and business entities), resources, and processes in a variety of forms to achieve the business goal. Most of the exchanged information may also include non-structured and non-transactional messages. The business logic is used to form service flow as parallel tasks or sequential tasks based on business rules, policies, and other business requirements.

Business processes represent the backbone of the flow of a business. The dynamic side of business architecture is realized through business processes. These business processes used to be implemented through a combination of static applications or, at best, a hardwired workflow. With service-orientation, a process can be realized by service compositions (which may be interim as an orchestration or a choreography) and the ability to insert “human intervention” and support long-running transactions.

In particular, compositions of services exposed in the Services Layer are defined in this layer: atomic services are composed into a set of composite services using a service composition engine. Note that composition can be implemented as a choreography of services or an orchestration of the underlying service elements.

Services combined or composed into flows or, for example, choreographies of services, bundled into a flow, work together to establish an application. These applications support specific use-cases and business processes. Typically, a visual flow composition tool will be used to design

the application flow. Figure 29 shows how a Business Process "P" can be implemented using Services A, B, C, and D from the Services Layer. *Process P* contains the logic for the sequence in which the services need to be invoked and executed, as well as having responsibility for many of the non-functional aspects such as state management. The services that are aggregated into a business process can be sourced from individual services or composite services.

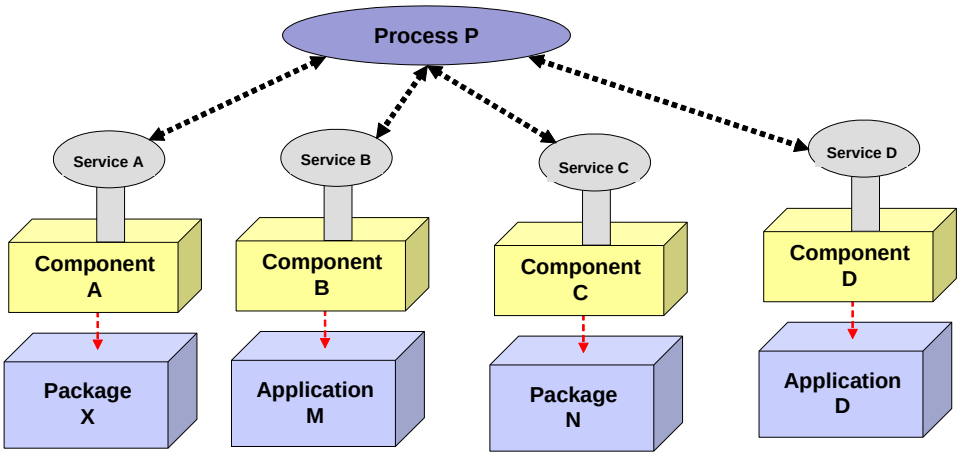


Figure 29: Services Orchestration

The Business Process Layer leverages the Services Layer to compose and choreograph services and to coordinate business processes to fulfill customer requirements. Visual flow composition tools such as BPMN-based tools can be used for design of application flow.

In more detail, the Business Process Layer performs three-dimensional process-level handling: top-down, bottom-up, and horizontal. From the top-down direction, this layer provides facilities to decompose business requirements into tasks comprising activity flows, each being realized by existing business processes, services, and service components. From the bottom-up direction, the layer provides facilities to compose existing business processes, services, and service components into new business processes. From the horizontal direction, the layer provides services-oriented collaboration control between business processes, services, and service components.

The decomposition of a business process first decomposes it into smaller tasks; then each task is mapped into coarse-grain service (i.e., candidate services) that will be realized by actual web services in the Services Layer. In other words, the layer provides the ability to decompose a business process into coarse-grain candidate services that fulfill the business functions.

Value Proposition

Building process blocks on-demand with reduced cost allows the supporting technology changes from being high volume/transactional to sophisticated but much smaller footprint applications. In fact, a business process captures the activities needed to accomplish a certain business goal. In today's business solutions, a business process has played a central role in bridging the gap between business and IT.

From the top-down approach, a business process can be defined by business people based on customers' requirements. In order to optimize the business process for better IT implementation, a business process should be componentized as re-usable services that can be modeled, analyzed, and optimized based on business requirements such as Quality of Service (QoS) (historical data described in the Quality of Service Layer), flow preference, price, time of delivery, and customer preferences. From the bottom-up approach, after a set of assets is created, they could be leveraged in a meaningful business context to satisfy customer requirements. The flexibility and extensibility of services composition guided by business requirements and composition rules enable business process on-demand for addressing different types of customer pinpoints by re-using services assets.

From an interaction perspective, the Business Process Layer communicates with the Consumer Layer (*aka* presentation layer) to communicate inputs and results with role players (e.g., end user, decision-makers, system administrator, etc.) through web portal or Business-to-Business (B2B) programs. Most of the control flow messages and data flow messages of the business process may be routed and transformed through the Integration Layer. The contents of the messages could be defined by the Information Layer. The Key Performance Indicators (KPIs) for each task or process could be defined in the Quality of Service Layer. The aggregation of services could be guided by the Governance Layer.

All the services should be represented and described by the Services Layer; and Service Components are represented by the Service Component Layer. From a technical perspective, dynamic and automatic business process composition poses critical challenges to researchers and practitioners in the field of web services. First, business processes are driven by business requirements, which typically tend to be informal, subjective, and difficult to quantify. Therefore, it is critical to properly formulate the descriptive and subjective requirements into quantifiable, objective, and machine-readable formats in order to enable automatic business process composition.

Second, existing web services-based business process description languages do not adequately accommodate detailed requirement specification, which fact makes it difficult to create optimal business process compositions.

Third, present web services specifications generally lack a facility to define comprehensive relationships among business entities, business services, and operations. These relationships may be important to optimize business process composition. For example, suppose enterprise E1 needs to compose a business process including service S. Enterprises E2 and E3 both provide similar service S.

However, there is a partnership between E1 and E2 that leads to a discount on services, and there is no partnership relationship between E1 and E3. If price is a requirement for party E1, this partnership between E1 and E2 needs to be taken into consideration in order to form the most appropriate business process. Fourth, nowadays more and more web services are published to the Internet on the daily basis. How to clearly specify search requirements to discover the most appropriate web services candidates remains a challenge. Fifth, a typical business process generally requires multiple web services to collaborate in order to serve business requirements.

Therefore, each Service Component not only needs to satisfy individual requirements, but also needs to coexist with other Service Components in order to best fit the overall composed

business process. In other words, the entire business process needs to be optimized prior to execution.

In summary, the Business Process Layer in the SOA RA plays a central coordinating role in connecting business-level requirements and IT-level solution components through collaboration with the Integration Layer, Quality of Service Layer, as well as the Information Layer, the Services Layer, and the Service Component Layer. Addressing those challenging issues are being covered in this Business Process Layer to further differentiate the proposed SOA RA with other conceptual reference models from other vendors.

Typical Flow

A typical calling sequence is to invoke a composite service in this layer, which implements a business process. This layer will then be responsible for orchestrating or choreographing the set of required underlying atomic or composite services that are combined to constitute the business process. It will typically maintain state for the flow, provide or collaborate with the Quality of Service Layer for monitoring the process flow, applying policies by working with the Governance Layer. Note that the invocation of the services may occur directly, or preferably, through the Integration Layer, thus enabling a separation of concerns between requester and provider to be managed by the capabilities and Architecture Building Blocks (ABBs) of the Integration Layer. For example, a single or federated set of Enterprise Service Bus(es) (ESBs) may be used to realize the invocation.

This layer will rely on the infrastructure provided by the Operational Systems Layer, in which, for example, the BPEL engine implementation will physically reside.

11.1.2 Capabilities

This layer supports and manages business processes and enables the SOA to choreograph or orchestrate services to realize business processes. Business Process Management (BPM) is to be found to start in this layer. There are multiple categories of capabilities that the Business Process Layer needs to support. These categories of capabilities are:

- **Process Definition**: This category of capabilities is required for defining the business processes/operational flow of the business.
- **Event Handling**: This category of capabilities handles business events in the context of a business process such as emitting/publishing events and subscribing/listening to business events.
- **Process Runtime Enablement**: This category of capabilities enables BPM and helps to realize the business processes in the runtime environment using standards such as BPEL, SCA, etc.
- **Process Information Management**: This category of capabilities manages the information needs of a business process such as managing its state, transforming data in the process flow, and maintaining a repository of assets.
- **Decision Management**: This category of capabilities defines and manages the decision points and associated rules within a business process.

- **Process Integration**: This category of capabilities facilitates integration with others layer of the SOA RA and helps to expose a business process as a service.
- **Security and Policy Compliance**: This category of capabilities enables access control and policy enforcement in the business processes.
- **Process Monitoring and Management**: This category of capabilities monitors and manages business processes, identifies bottlenecks in the business processes, and optimizes workload assignment.

This layer features the following capabilities:

Process Definition

1. Ability to define business processes representing dynamic behavior of the business

Event Handling

2. Ability to detect, emit, and listen to business events in the context of business processes

Process Runtime Enablement

3. Ability to realize and deploy the business processes in a runtime environment
4. Ability to create and manage individual instances of the business processes
5. Ability to execute the instances of a business process, its sub-processes, and activities therein
6. Ability to define the elements of an assembly at design time and have the assembly occur at runtime based on a set of rules
7. Ability to intelligently decide the end-point of the enabling services using the process context
8. Ability to manage the interaction of the business process with humans

Process Information Management

9. Ability to manage the context of a business process
10. Ability to manage the state of a process
11. Ability to transform the data flowing through a business processes based on its needs
12. Ability to store and retrieve assets required and requested by an ongoing process

Decision Management

13. Ability to configure the relationships between the composition and the non-functional characteristics of the process flow

14. Ability to encapsulate/isolate the decisions and rules affecting those decisions associated with the execution of a business process from the actual process flow itself

Process Integration

15. Ability to make available the business processes as a service

16. Ability to schedule the execution of a business process

Security and Policy Compliance

17. Ability to define policies, enforce them, and verify the compliance of the elements of process with a set of predefined policies

18. Ability to control access to a process flow at design or runtime

Process Monitoring and Management

19. Ability to monitor a business process and insert points at which metrics may be gathered, identify bottlenecks, and optimize workload assignments

11.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Business Process Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Process Definition	Business Process	1
2	Event Handling	Integration Layer: Event Listener	2
3		Integration Layer: Event Producer	2
4	Process Runtime Enablement	Process Container/Process Engine	3, 4, 5
5		Process Manager	3, 4, 5
6		Process Factory	4, 5
7		Process Flow Manager	4, 5
8		Process Controller	4, 5
9		Dynamic Assembler	6, 7
10		Workflow Manager/Human Task Manager	8
11	Process Information Management	Context Manager	9
12		Process State Manager	10

#	Capability Category	ABB Name	Supported Capabilities
13		Integration Layer: Data Transformer	11
14		Process Repository	12
15	Decision Management	Governance Layer: Business Rules Manager	13, 14
16	Process Integration	Process Service Adapter	15
17		Scheduler	16
18	Security & Policy Compliance	Quality of Service Layer: Policy Enforcer	17
19		Quality of Service Layer: Access Controller	18
20	Process Monitoring and Management	Quality of Service Layer: Business Activity Monitor	19

Table 4: ABB to Capability Mapping for the Business Process Layer

11.2 Details of ABBs and Supported Capabilities

11.2.1 Details of ABBs

This section describes each of the ABBs in the Business Process Layer in terms of their responsibilities.

11.2.1.1 Business Process

This ABB represents a process that a business performs. This ABB is one of the core fundamental ABBs in the SOA RA. Business process is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm.

11.2.1.2 Integration Layer: Event Listener

See Event Listener ABB in the Integration Layer.

11.2.1.3 Integration Layer: Event Producer

See Event Producer ABB in the Integration Layer.

11.2.1.4 Process Container or Process Engine

This ABB provides an environment to manage the execution and flow of business processes, and to manage the interaction of humans with business processes. It is also is responsible for managing the instances of running processes and their context.

11.2.1.5 Process Manager

This ABB is responsible for deploying processes in the process container.

11.2.1.6 Process Factory

This ABB is responsible for creating instances of deployed processes in the process container.

11.2.1.7 Process Flow Manager

The ABB is responsible for managing process templates and process instances. Process templates describe the business process model. At runtime, this ABB creates process instances from process templates using the Process Factory ABB. A process instance is the representation of a single running business process. The ABB's job is to manage one or more process instances concurrently. It can support multiple interfaces that interact with business processes; for example, an SCA interface for interacting with other service components, and a generic process API for interacting with J2EE clients. It works cooperatively with the Workflow Manager or Human Task Manager ABB. The ABB has a core responsibility to split and manage processes and its sub-processes together and enabling services. This becomes an important part of an SOA's ability to support business processes of varying granularity. For example, a *Process Claim* business process might have underlying business processes that support the *Process Claim* business process. This ABB in this case then manages all the interactions and the decomposition relationships for that individual's sub-processes and enabling services.

11.2.1.8 Process Controller

The ABB acts as a controller that supports all the interactions between the invocation of a business process and the building blocks supporting that invocation. It is the core of a process container and responsible for executing the process activities.

11.2.1.9 Dynamic Assembler

This ABB is responsible for invoking the appropriate end-point that needs to serve the request based on the context. Context provides the extra information that this ABB needs to make the intelligent decisions on which end-point is to be invoked.

11.2.1.10 Workflow Manager or Human Task Manager

This ABB is responsible for the coordination of requests that require human intervention. It is also responsible for the management of the state of human tasks and work items. This ABB supports the ability of the Business Process Layer to integrate human intervention into business processes. In practice this might mean using the Service Adapter and Integration Adapter to integrate with the Consumer Layer. This ABB is often required in process error handling situations (e.g., the involvement of a customer service representative when the customer enters a wrong account number in a self-service banking scenario).

11.2.1.11 Process State Manager

The ABB supports the ability of the layer to retain and manage state (not status) during a business process. It manages the process state within each business process instance. This is an important capability, for example, for orchestrating a series of state transitions which might involve multiple business processes.

11.2.1.12 Context Manager

The ABB manages the context of various instances of the business process.

11.2.1.13 Integration Layer: Data Transformer

See Data Transformer ABB in the Integration Layer.

11.2.1.14 Process Repository

The ABB acts a repository store for all business processes. This ABB is internal to the layer and interfaces with the Data Repository ABB in the Information Layer and Asset Repository ABB in the Governance Layer. It is the Process Repository ABB where other ABBs such as the Process Controller ABB will turn to obtain process information such as state information.

11.2.1.15 Process Service Adapter

This ABB integrates the Business Process Layer with other SOA RA layers, in particular the Services, Integration, and Consumer Layers. It acts as the mechanism to expose business processes as services.

11.2.1.16 Scheduler

This ABB supports the ability to schedule the invocation of business processes and process activities at different times.

11.2.1.17 Governance Layer: Business Rules Manager

See Business Rules Manager ABB in the Governance Layer.

11.2.1.18 Quality of Service Layer: Policy Enforcer

See Policy Enforcer ABB in the Quality of Service Layer.

11.2.1.19 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

11.2.1.20 Quality of Service Layer: Business Activity Monitor

See Business Activity Monitor ABB in the Quality of Service Layer.

11.2.2 Structural Overview of the Layer

The Business Process Layer is a critical component of the SOA RA. ABBs in the Business Process Layer can be thought of as being logically partitioned into categories which support:

- Definition, composition, and decomposition of a business process
- Handling of events
- Runtime enablement and realization of business processes

- Management of information and associated data flow in the context of the business processes
- Management of decision points/points of variability and associated rules in the context of the business processes
- Integration capabilities necessary for realizing business processes
- Security and policy compliance pertaining to business processes
- Monitoring and management of business processes

Figure 30 illustrates the ABBs partitioned into key categories:

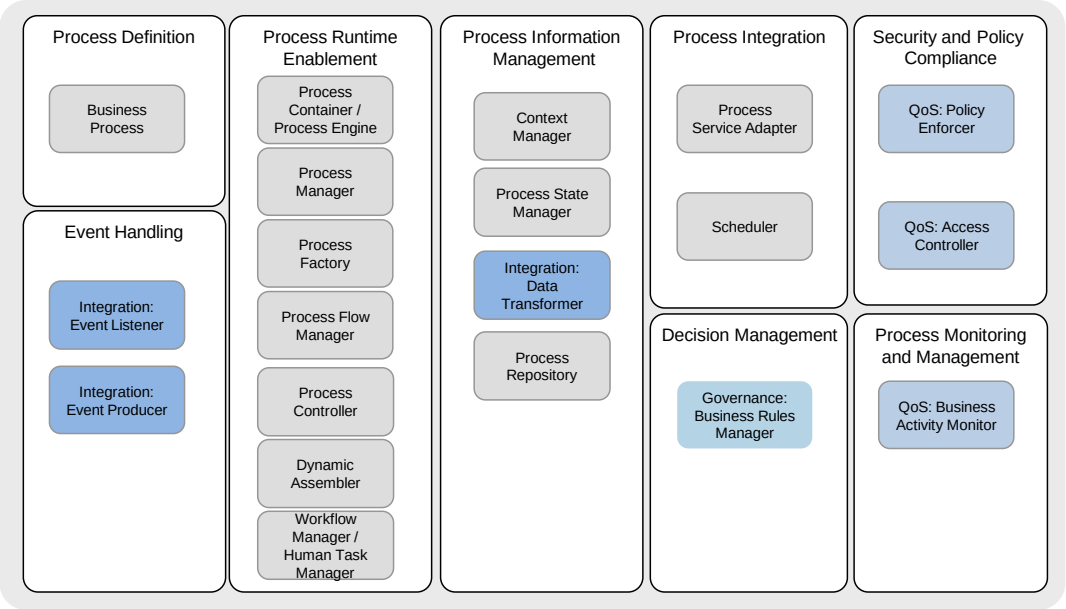


Figure 30: ABBs in the Business Process Layer

Each business process is composed of data flows and process/control flows. Data flows pertain to how information is represented and conveyed in relation to the process data flow. The process and control flow pertains to the sequence of activities or services that are being invoked to enact a business process.

11.3 Inter-Relationships between the ABBs

Figure 31 show the inter-relationships and a non-normative sequence of interaction. The final, prescriptive sequence of interaction will be defined by the underlying solution architecture and the standards that are invoking the support of other SOA RA layers.

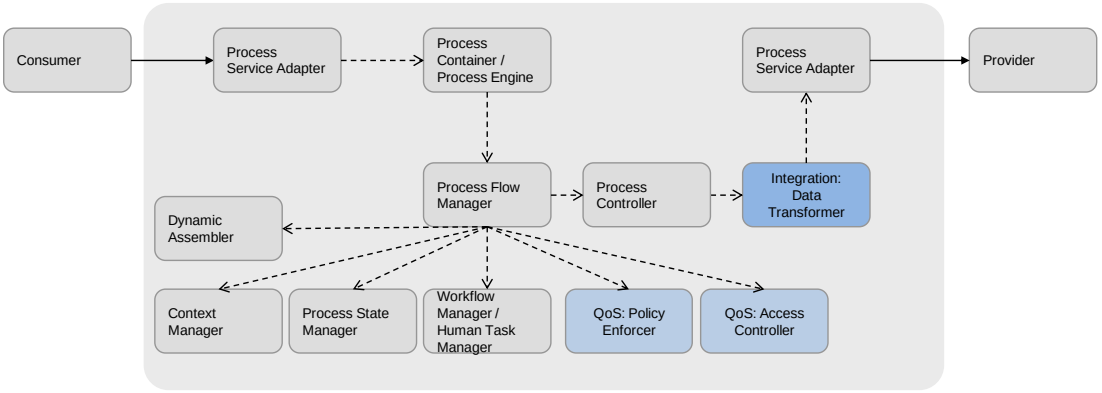


Figure 31: Key Relationships among ABBs in the Business Process Layer

11.4 Significant Intersection Points with other Layers

11.4.1 Interaction with Cross-Cutting Layers

The Business Process Layer relies on cross-cutting layers of the architecture to fulfill its responsibilities.

It relies on the Governance Layer for the following capabilities:

- Ability to store metadata for policies
- Ability to support management (storage, retrieval, etc.) of rules to support rules associated with decision points in service mediation, orchestration, and composition

It relies on the Quality of Service Layer for the following capabilities:

- Ability to authenticate/authorize for service invocation

It relies on the Information Layer for the following capabilities:

- Ability to store and retrieve metadata and data required for the execution of the business processes

It relies on the Integration Layer for the following capabilities:

- Ability to invoke services to realize systematic process steps
- Ability to transform data from one format to another

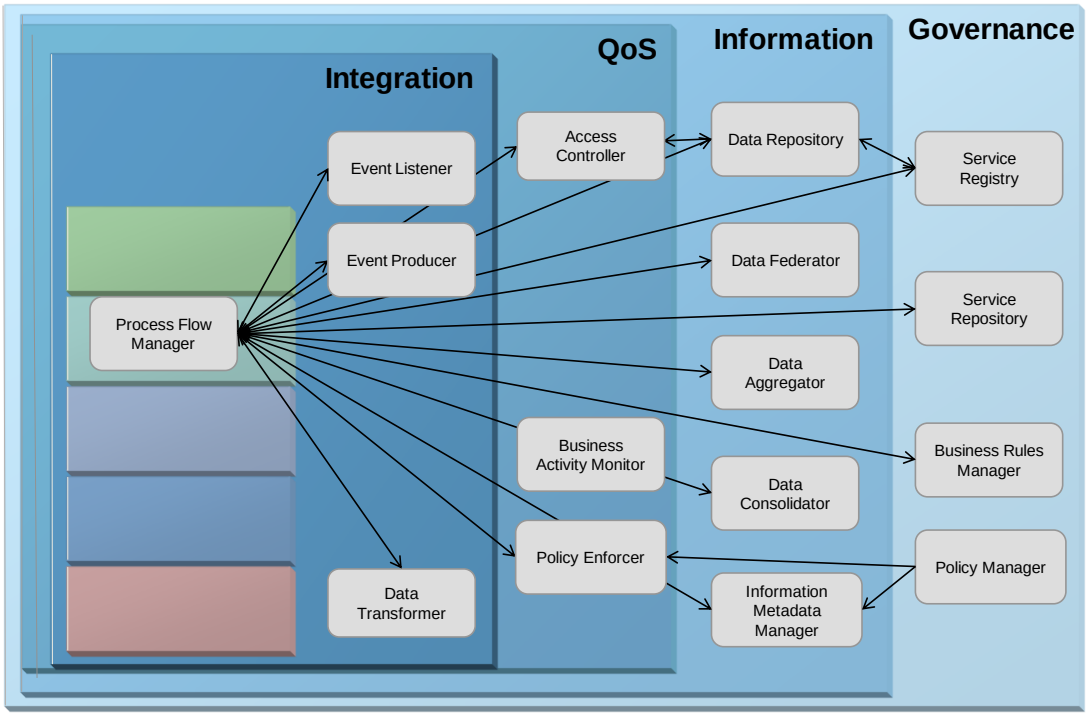


Figure 32: Key Interactions of the Business Process Layer with Cross-Cutting Layers

Therefore, Business Process Layer interfaces with the following ABBs of cross-cutting layers of the architecture to provide its capabilities:

- It leverages the Service Registry ABB and Service Repository ABB from the Governance Layer for storing metadata such as policy, schema, etc. and for providing access to the metadata. This Service Registry ABB also contains service definitions at runtime and supports service virtualization and service discovery. Service virtualization in this context is the exposure of a service end-point through a "proxy" (the registry).
- It leverages the Business Rule Manager ABB in the Governance Layer to manage the rules supporting the decision points or points of variability within a business process.
- It leverages the Access Controller ABB in the Quality of Service Layer for authenticating/authorizing the facility for service invocation and workload assignment.
- It leverages the Data Aggregator ABB, Data Federator ABB, Data Consolidator ABB, Information Metadata Manager ABB, and Data Repository ABB from the Information Layer to store and assess data.
- It leverages the Access Controller ABB in the Quality of Service Layer to enforce access privileges and the Policy Enforcer ABB in the Quality of Service Layer to enforce policies.
- It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Layer such as data transformation, service request, etc. It leverages the Mediator ABB in the Integration Layer to integrate with existing systems and applications. It leverages the Data Transformer ABB in the Integration Layer to transform

data from one format to other. It leverages the Event Producer ABB and Event Listener ABB in the Integration Layer to publish events or subscribe to events.

11.4.2 Interaction with Horizontal Layers

The Consumer Layer can invoke or trigger events that will execute business processes in the Business Process Layer. Business processes in the Business Process Layer are composed of services in the Services Layer either using orchestration or choreography. Essentially an executable business process is composed of either human task or systematic steps. The systematic steps can be enabled by services in the Services Layer.

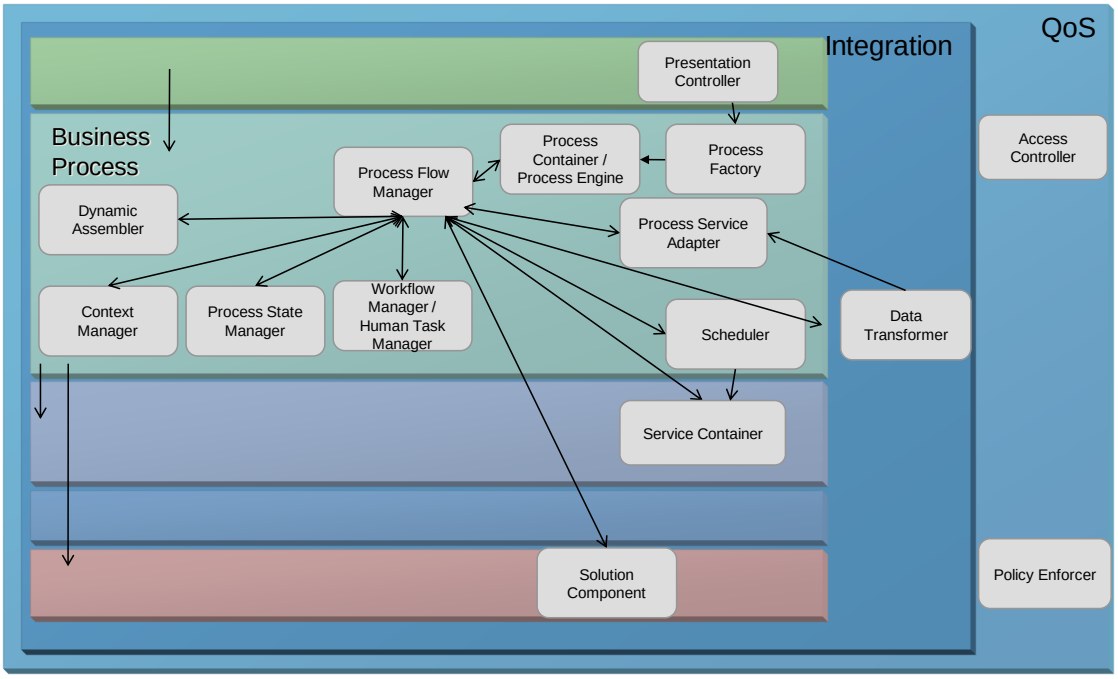


Figure 33: Key Interactions of the Business Process Layer with Horizontal Layers

11.5 Usage Implications and Guidance

The Business Process Layer incorporates the ability to either statically or dynamically configure the orchestration or choreography of services in its composition.

To summarize, the Business Process Layer supports capabilities required for enabling SOA such as process composition and decomposition, orchestration or choreography of services, collaboration between processes, information and services using a set of policies and business rules that aid in its execution of the process flow.

12 Consumer Layer

12.1 Overview

12.1.1 Context and Typical Flow

The Consumer Layer is the point where consumers interact with the SOA. It enables an SOA to support a client-independent, channel-agnostic set of functionality, which is separately consumed and rendered through one or more channels (client platforms and devices). Thus, it is the point of entry for interactive consumers (humans and other applications/systems) and services from external sources (e.g., Business-to-Business (B2B) scenarios).

In fact the Consumer Layer is the entry point for all external consumers, *external to the SOA*. This can be other systems, other SOAs, cloud service consumers, human users, etc.

This decoupling between the consumer and the rest of the underlying SOA provides organizations with the ability to support agility, enhanced re-use, and improve quality and consistency. Channels can be thought of as the platforms by which SOA consumers access services through the SOA. Examples of channels would be web front-ends, and IVR (Interactive Voice Response) systems, which could both leverage the same core functionality within the SOA.

It is thus important to note that SOA decouples the user interface (channel), and thus the consumer, from the components and the implementation of the functionality. Examples of this include Service Component Architecture (SCA) Components [10], portlets, WSRP (Web Services for Remote Portlets 2.0) [14], and B2B integration interfaces. Thus, the SOA enables us to cater for human interactive interfaces (user interfaces) as well as system (application or software) consumers.

The Consumer Layer is that part of the SOA which enables channel-independent access to business processes and services supported by various applications and platforms. This is important for the effective use and adoption of the SOA.

Thus, the Consumer Layer provides the capabilities required to deliver IT functions and data to end users and as the entry point for consumers into the SOA RA. These capabilities enable specific users to customize preferences, integrate with consumer channels, including rich clients (mashups and Ajax [20]) and act as a mechanism for the underlying SOA to expose its functionality. Standards (such as WSRP) may leverage services at the application interface or presentation level. Its capabilities include the ability to quickly create the front end of the business processes and the composite applications to respond to changes in the market. It provides the point where in-bound consumer requests have security and other Quality of Service (QoS) policies asserted to ensure that the request is secure and brought into the context of the SOA.

The Consumer Layer provides the ability to integrate services from within the SOA, and the ability to transform, integrate, and personalize information into the content and mediate with the consumer channels (both for user and non-user interfaces).

12.1.2 Capabilities

There are multiple categories of capabilities that the Consumer Layer needs to support in the SOA RA. These categories are:

- **Consumer Services**: This category of capabilities addresses the support of interaction with consumers.
- **Presentation Services**: This category of capabilities addresses the support of presentation services, which include a presentation, composite view and presentation control, and the consumer-centric configuration of views.
- **Backend Integration**: This category of capabilities addresses the integration of the Consumer Layer with backend and legacy systems using SOA and services and transforms their information and incorporates it into content.
- **Caching and Streaming Content**: This category of capabilities addresses the support of information buffering and performance, and supports the operation of the Consumer Layer.
- **Security and Privacy**: Capabilities that address the support of QoS, information protection, and security.
- **Information Access**: This category of capabilities addresses the sharing of data and metadata across the layers of the SOA RA such as QoS attributes, attributes defining common rules to be used across the layers, etc.

This layer features the following capabilities:

Consumer Services

1. Ability to consume (use) the SOA, through a program or an individual who requests a service
2. Ability to support consumer interaction and integration; i.e., the ability to capture the input from the user (consumer) of the SOA and provide the response to the consumer

Presentation Services

3. Ability to support the creation of a presentation view by the composition of a number of atomic components
4. Ability to configure information which will support specific capabilities associated with ensuring consistency (similar to a style guide)
5. Ability to provide navigation logic and flow for the processing of consumer interactions (presentation control)

6. Ability to provide the Consumer Layer with the ability to support customer-specific information (enabled by the Information Layer) and personalization and customer-specific preferences to be used by the presentation controller for navigation and content presentation purposes

7. Ability to configure components in the Consumer Layer based on consumer request scenarios

Backend Integration

8. Ability to mediate services from other SOA layers such as the Business Process Layer and the Integration Layer into the Consumer Layer; it provides the ability to integrate the underlying SOA into the Consumer Layer

9. Ability to support the translation of input data/content from a format supported by the user of the SOA to a format required by the other layers of the SOA and to convert content returned from them into a user acceptable response format

Caching and Streaming Content

10. Ability includes the handling of streaming content

11. Ability to cache interaction data to improve performance and quality

Security and Privacy

12. Ability to provide access to authentication/authorization capabilities (enabled through policies) to be used by the presentation controller to allow/prevent what content can be presented to the consumer

13. Ability to filter to control access to the underlying SOA

14. Ability to monitor the usage of the Consumer Layer components

Information Access

8. Ability to access data and metadata through the Information Layer

12.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Consumer Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Consumer Services	Consumer	1
2		Client (Channel)	2
3	Presentation Services	Presentation Adapter	4
4		Presentation Controller	3, 6, 7
5		Presentation Flow Manager	5

#	Capability Category	ABB Name	Supported Capabilities
6		Composite View	3
7		Consumer/User Profile Manager	4, 6
8		Personalization Manager	6
9	Backend Integration	Integration Layer: Integration Controller	8
10		Integration Layer: Data Transformer	9
11	Caching and Streaming Content	Cache	10-11
12	Security and Privacy	Governance Layer: Policy Manager	12

Table 5: ABB to Capability Mapping for the Consumer Layer

12.2 Details of ABBs and Supported Capabilities

12.2.1 Details of ABBs

This section describes each of the ABBs in the Consumer Layer in terms of their responsibilities.

12.2.1.1 Consumer

This ABB is the individual consumer (actor) that uses the services supported by the SOA RA. The consumer can be human or a system.

12.2.1.2 Client (aka Channel)

This ABB interacts with the Presentation Controller ABB to use the underlying services, integrating the consumer of services supported by the SOA RA. It is the element in the SOA which the consumer interacts with. As such, it provides the point interaction for the consumer. The key responsibilities include dealing with the nature of interaction that the client has with the consumer in terms of:

- What data is provided?
- What will be the format?
- What will be the interaction?

Examples of clients may be IVRs, rich-clients (JSF, Ajax), mobile applications, etc. It will be where Web 2.0 client support will go (for example, the ability to create mashups). It is also the component that *renders* the view to the consumer.

12.2.1.3 Presentation Adapter

This ABB is responsible for integrating the client with the rest of the Consumer Layer. It accepts client-specific information and separates client-specific standards from the rest of the Consumer Layer, transforming data into formats that the rest of the Consumer Layer understands.

12.2.1.4 Presentation Controller

This ABB is responsible for handling the orchestration, decomposition, and composition of the view rendered by the client. It uses the Presentation Flow Manager ABB, Composite View ABB, and other ABBs to create the view and to submit requests to retrieve data from other layers in the SOA RA.

12.2.1.5 Presentation Flow Manager

This ABB is responsible for supporting navigation and control flow in the Consumer Layer. It is an important part in the assemblage of a view component to send back and render in the client.

12.2.1.6 Composite View

This ABB is responsible for assembling data received from various services and creates a composite view which is orchestrated and then passed on to the client for rendering.

12.2.1.7 Consumer/User Profile Manager

This ABB is responsible for supporting the personalization of the interface and the presentation to a particular consumer's wants and needs. It will be used both by the Client ABB and Presentation Controller ABB. It can be used for controlling individual consumer features or the creation of profiles based on roles.

12.2.1.8 Personalization Manager

This ABB allows users to customize the appearance of the user interface according to personal preferences. The customization is accomplished partly through administrative set-up, which defines the default settings and access rights to user interfaces/web pages. It supports both:

- Rule-based personalization to select content for the user such as a rule might display special discounts to gold customers, but only during the summer months
- Collaborative filtering technology-based personalization to select content based on common interests or behaviors

12.2.1.9 Integration Layer: Integration Controller

See Integration Controller ABB in the Integration Layer.

12.2.1.10 Integration Layer: Data Transformer

See Data Transformer ABB in the Integration Layer.

12.2.1.11 Cache

This ABB is responsible for supporting the scalability of the layer and supporting the caching of interaction data to improve performance.

12.2.1.12 Governance Layer: Policy Manager

See Policy Manager ABB in the Governance Layer.

12.2.1.13 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

12.2.1.14 Information Layer: Data Repository

See Data Repository ABB in the Information Layer.

12.2.2 Structural Overview of the Layer

The ABBs in the Consumer Layer can be thought of as being logically partitioned into categories which support:

- Ability to support the interaction of the SOA with consumers

Ability to support presentation, the creation of composite views and presentation control, content composition and decomposition, and the consumer-centric configuration of views

- Ability to integrate information from services from the SOA and transform their information and incorporate it into content[4]
- Ability to support information buffering and performance, and support the operation of the Consumer Layer
- Ability to support QoS, information protection, and security
- Ability to address the sharing of metadata across the layers of the SOA RA

[4] Content here refers to any information returned to the consumer. This can be text, images, data, etc. What it physically is, is very solution-specific and will also vary based on the kind of consumer.

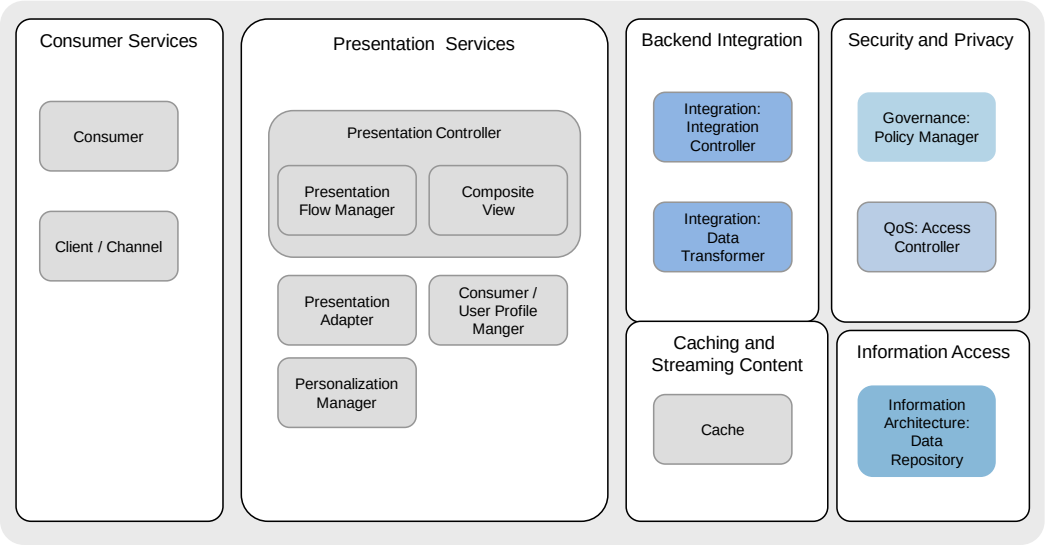


Figure 34: ABBs in the Consumer Layer

12.3 Inter-Relationships between the ABBs

In the Consumer Layer, there is a logical partitioning between integration with the consumer, integration with the services components of the SOA, integration with some of the cross-cutting SOA layers (information and QoS), and the functionality for the creation of service invocations (requests) to the SOA and composing the returned and cached content to the consumer. Consumers can be human or system. The scenarios below illustrate usage of the Consumer Layer. The first scenario shows the interaction with human service consumers initiating the interaction with the SOA. The second scenario shows the interaction with systematic (non-human) service consumers initiating the interaction with the SOA. It is important to note that practically there is no real difference between human and non-human actors – they all represent interactions with the SOA. The extent of differences between the two scenarios really is driven by the nature of the interaction and the channel, and is thus very solution-specific. These examples have been provided as illustrative examples. The third scenario is a multi-channel scenario illustrating a typical environment involving both human and other actors. As it shows, the Consumer Layer provides the separation of concerns to enable the SOA to have maximum re-usability and agility, leveraging the core services in the Integration Layer and Business Process Layer and the capabilities of the rest of the SOA.

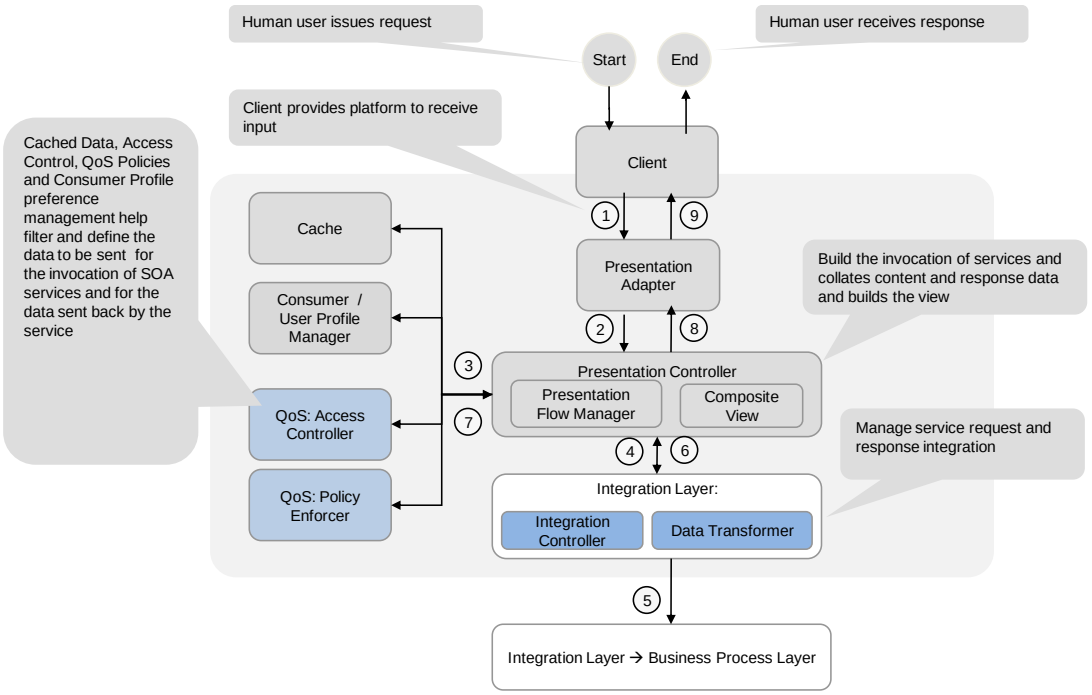


Figure 35: Interaction of a Human Service Consumer with the SOA via the Consumer Layer

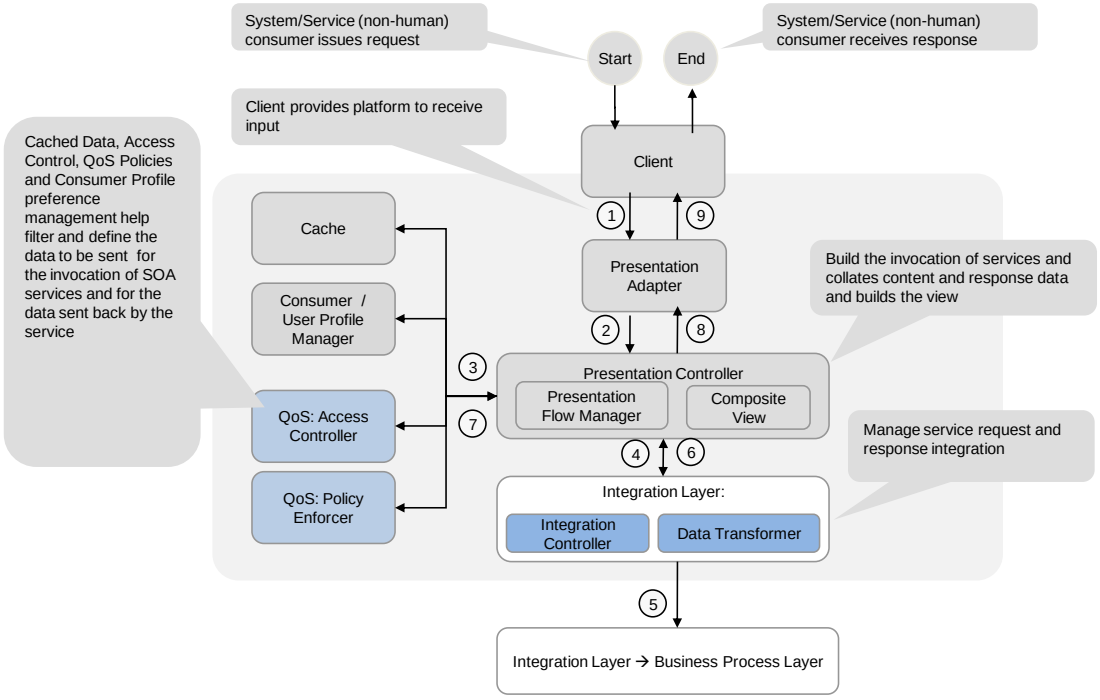


Figure 36: Interaction of a Systematic (Non-Human) Service Consumer with the SOA via the Consumer Layer

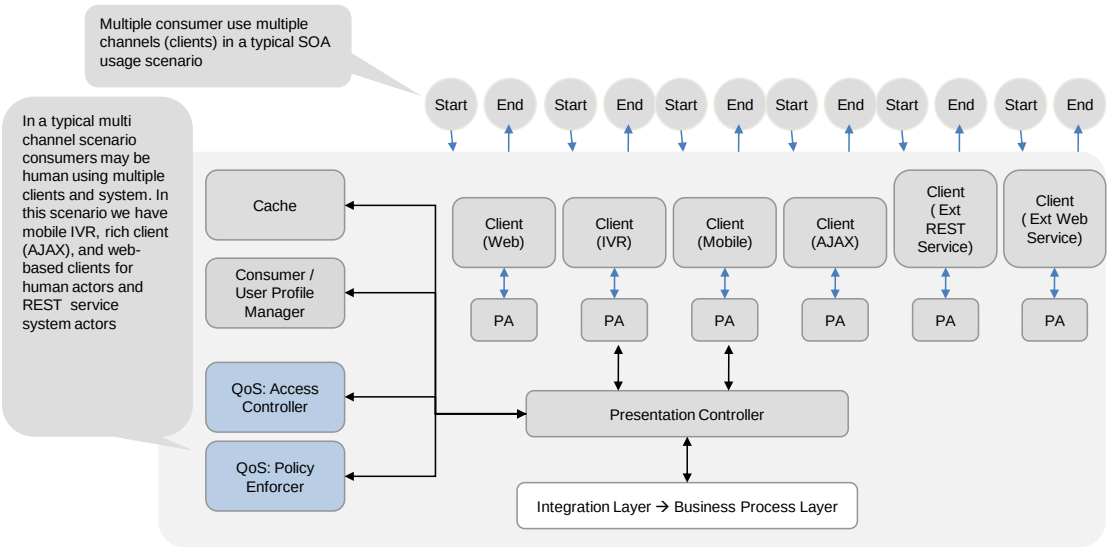


Figure 37: A Typical SOA Usage Scenario with Multiple Consumers using Multiple Channels

Figure 37 shows multiple entry points (consumers) making requests using multiple clients. Each channel has a unique platform/client and needs to be integrated using specific instantiations of the presentation adapter (which is why we show multiple instances of the ABBs).

12.4 Significant Intersection Points with other Layers

12.4.1 Interaction with Cross-Cutting Layers

The Consumer Layer relies on cross-cutting layers of the architecture to fulfill its responsibilities.

It relies on the Governance Layer for the following capabilities:

- Ability to store metadata for policies

It relies on the Quality of Service Layer for the following capabilities:

- Ability to authenticate/authorize for service invocation

It relies on the Information Layer for the following capabilities:

- Ability to store and retrieve metadata and data

It relies on the Integration Layer for the following capabilities:

- Ability to invoke business processes and/or services
- Ability to transform data from one format to another

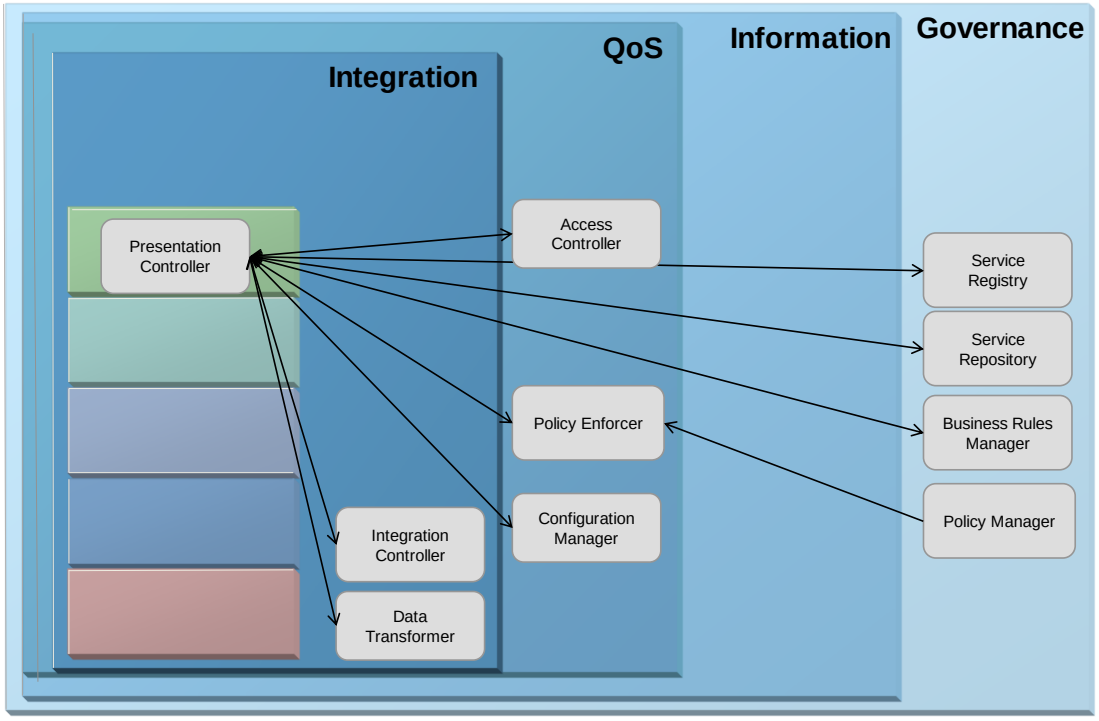


Figure 38: Key Interactions of the Consumer Layer with Cross-Cutting Layers

Therefore, the Consumer Layer interfaces with the following ABBs of cross-cutting layers of the architecture to provide its capabilities:

- It leverages the Access Controller ABB and Policy Enforcer ABBs in the Quality of Service Layer to enforce access control privileges and other policies.
- It leverages the Data Aggregator ABB, Data Federator ABB, Data Consolidator ABB, Information Metadata Manager ABB, and Data Repository ABB from the Information Layer to store and assess data.
- It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Layer such as data transformation, service request, etc. It leverages the Mediator ABB in the Integration Layer to integrate with existing systems and applications. It leverages the Data Transformer ABB in the Integration Layer to transform data from one format to other. It leverages the Event Producer ABB and Event Listener ABB in the Integration Layer to publish events or subscribe to events.

12.4.2 Interaction with Horizontal Layers

Within the Consumer Layer, the client/channel primarily interacts with the Consumer Profile Manager ABB and the Presentation Controller ABB and also contains configuration information for the other ABBs in the Consumer Layer. The Presentation Controller ABB interacts with the Integration Controller to provide support for the interaction of the services in the Services Layer being used by the consumer. In addition, the Presentation Controller ABB may work with the Data Transformer to map data from native formats offered from the existing operational systems in the services to the formats needed to support the client/channel. Consumers interact with the

Service Registry and Service Repository to get the service description information needed to support the interaction with the service.

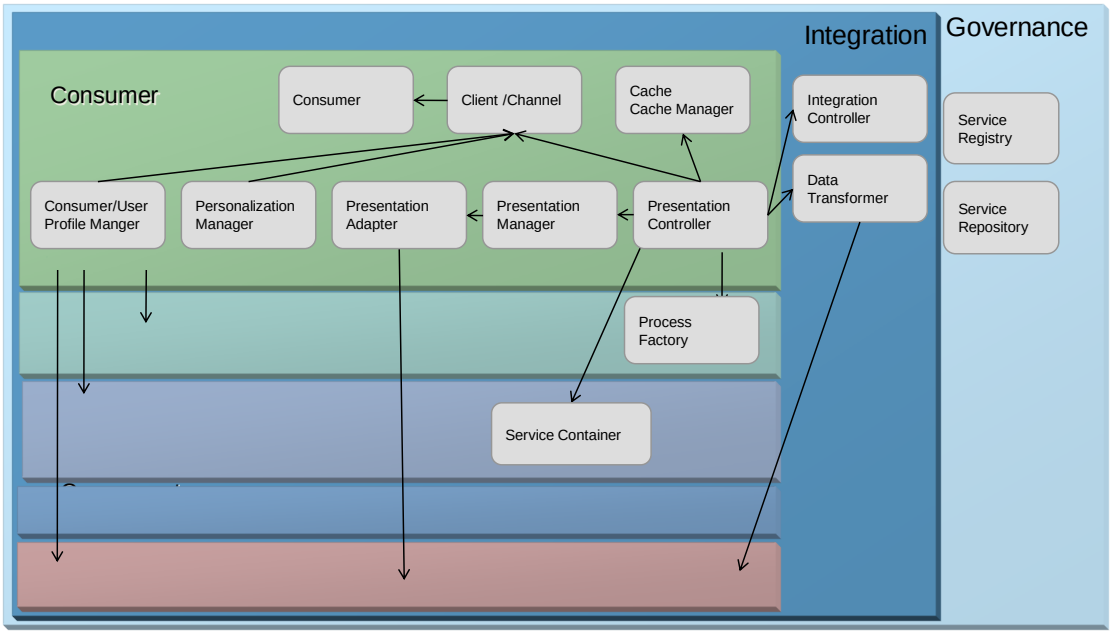


Figure 39: Key Interactions of the Consumer Layer with Horizontal Layers

12.5 Usage Implications and Guidance

This layer provides the SOA with a point of integration between consumer requests and the underlying SOA. It separates dependencies from how the services are implemented and what the consumers are. Standards such as WSRP enable web service integration, encapsulating users, and letting different SOA solutions be used. The architecture lets organizations and industry organizations maintain consistent standards and common implementations.

13 Integration Layer

13.1 Overview

13.1.1 Context and Typical Flow

The Integration Layer is a key enabler for an SOA as it provides the capability to mediate which includes transformation, routing, and protocol conversion to transport service requests from the service requester to the correct service provider. Thus, it supports the capabilities required for enabling SOA such as routing, protocol support and conversion, messaging/interaction style, support for heterogeneous environment, adapters, service interaction, service enablement, service virtualization, service messaging, message processing, and transformation.

It is the layer in the SOA RA that supports the integration with solution platforms by the other layers in the SOA RA using "adapters", the access of services by other layers, and the capabilities associated with service transport. It can be thought of as the plumbing which interconnects the SOA.

This layer enables the service consumer/requestor to connect to the correct service provider through the introduction of a reliable set of capabilities. The integration can start with modest point-to-point capabilities for tightly-coupled end-points and cover the spectrum to a set of much more intelligent routing, protocol conversion, and other transformation mechanisms often described as, but not limited to, an Enterprise Service Bus (ESB). WSDL specifies a binding, which implies location where a service is provided, and is one of the mechanisms to define a service contract. An ESB, on the other hand, provides a location-independent mechanism for integration, and service substitution or virtualization.

The integration that occurs here is primarily the integration of Layers 2 through 4 (the "functional" *aka* horizontal layers of the SOA RA). For example, this is where binding (late or otherwise) of services occurs for process execution. This allows a service to be exposed consistently across multiple customer-facing channels such as web, IVR, XML client, etc. The transformation of response to HTML (for web), VoiceXML (for IVR), XML String, can be done via XSLT functionality supported through a message transformation capability in the Integration Layer.

As shown in Figure 40, the Integration Layer:

- Provides a level of indirection between the consumer of functionality and its provider. A service consumer interacts with the service provider via the Integration Layer. Hence, each service interface is only exposed via the Integration Layer (e.g., ESB), never directly and point-to-point integration is done at the Integration Layer instead of consumers/requestors doing it themselves.
- Consumers and providers are decoupled; this decoupling allows integration of disparate systems into new solutions.

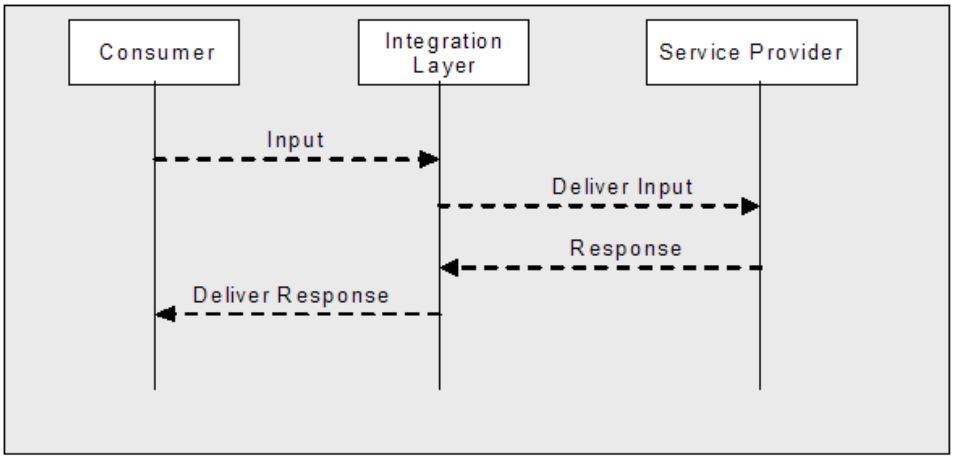


Figure 40: Usage of the Integration Layer

13.1.2 Capabilities

There are multiple set of categories of capabilities that the Integration Layer needs to support in the SOA RA. These categories are:

- **Communication, Service Interaction, and Integration**: This category of capabilities provides the ability to route requests to correct the provider after necessary message transformation and protocol conversion and to connect the service requestor to the service provider and its underlying solutions platforms realizing the requested service. It also provides the ability to discover services and, at runtime, to support the virtualization of services so that changes to the end-points (or locations from where the services are called and where the services are provided) can occur without impact to service consumers and service providers.
- **Message Processing**: This category of capabilities provides the ability to perform the necessary message transformation to connect the service requestor to the service provider and to publish and subscribe messages and events asynchronously.
- **Quality of Service**: This category of capabilities supports handling of transactions and exceptions and other NFRs.
- **Security**: This category of capabilities helps in enforcement of access privileges and other security policies.
- **Management**: This category of capabilities provides the ability to maintain service invocation history and monitor and track the service invocations.

This layer features the following capabilities:

Communication, Service Interaction, and Integration

1. Ability to take a service call and messages to the end-point; i.e., to enable a service consumer to connect/interact with service providers
2. Ability to handle service request and service response

3. Ability to support communication through a variety of protocols
4. Ability to support variety of messaging styles such as one-way, pub-sub, request-response
5. Ability to route messages to the correct service provider
6. Ability to transform protocol formats; e.g., from SOAP/HTTP to SOAP/Message Queue or SOAP/JMS
7. Ability to link a variety of systems that do not directly support service-style interactions so that a variety of services can be offered in a heterogeneous environment
8. Ability to store and forward messages using message queuing

Message Processing

9. Ability to transform data formats; e.g., from proprietary to standard format or industry standards and *vice versa*
10. Ability to transform semantic mapping (data positional mapping)
11. Ability to aggregate (including messages and data) from different services and service providers
12. Ability to propagate the events from producers to consumers

Quality of Service

13. Ability to handle transactions from the other layers, especially when a statically composed service invokes a service chain
14. Ability to handle exceptions raised in the process of service invocation and message passing

Security

15. Ability to authenticate/authorize for service invocation and message routing

Management

16. Ability to capture and record message routing and service invocation history
17. Ability to track and monitor the message routing and service invocation activities
18. Ability to configure the Integration Layer

13.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Integration Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Communication, Service Interaction and Integration	Integration Controller/ Integration Gateway	1-7
2		Adapter	1, 2, 3, 6, 7
3		Asynchronous Messaging Manager	4, 8
4		Mediator	1-7
5		Router	5
6		Protocol Converter	6
7	Message Processing	Message Transformer	9, 10
8		Data Transformer	9
9		Semantic Transformer	10
10		Data Aggregator	11
11		Event Broker	12
12		Event Producer	12
13		Event Listener	12
14	Quality of Service	Transaction Manager	13
15		Exception Handler	14
16	Security	Quality of Service Layer: Access Controller	15
17	Management	Logger	16
18		Auditor	17
19		Quality of Service Layer: Configuration Manager	18

Table 6: ABB to Capability Mapping for the Integration Layer

13.2 Details of ABBs and Supported Capabilities

13.2.1 Details of ABBs

This section describes each of the ABBs in the Integration Layer in terms of their responsibilities.

13.2.1.1 Integration Controller/Integration Gateway

This ABB serves as an entry point to this layer. Other layers interact with this ABB to leverage other ABBs in this layer to fulfill their respective responsibilities. In turn, this ABB is thus responsible for interfacing with other ABBs in this layer and managing the interaction flow among the ABBs in this layer. For example, it delegates to the Router ABB to service requests

and it delegates to the Message Transformer ABB for the data, format, and message transformation needs of the other layers.

13.2.1.2 Adapter

This ABB is responsible for the interfacing/connectivity of SOA RA layers of a solution to external systems and components and taking a call (message) to the end-point. In particular, it addresses any necessary mediation (router, message transformer, protocol converter) of the elements outside the Integration Layer and the interaction with external systems and components. This ABB should typically be used by ABBs in all SOA RA layers to access external components of solution platforms, providing a consistent integration capability.

13.2.1.3 Mediator

This ABB is responsible for handling the service request/response interaction. It also supports the transformation between message formats, conversion of protocols, and routing of service call/messages to the service provider. It uses the Data Transformer ABB and optionally the Semantic Transformer ABB for the transformations. Finally, it supports static service composition to orchestrate and chain services or system calls or messages together in order to compose services statically. It uses ABBs such as Router, Message Transformer, and the Data Aggregator to do this.

13.2.1.4 Router

This ABB is responsible to route messages between service consumer/requestor and service provider including those based on both content-based routing, straight through message passing. It may change the route of a message, selecting among service providers that support the intent of the requester. Selection criteria for the provider can include content and context of the message as well as knowledge about capabilities of the target candidates and even versioning of service implementation. If a message is routed to one provider, which responds with a failure, the message can be re-routed to another provider. In certain circumstances it may be used to route messages without the involvement of the Mediator ABB to realize straight message pass-through. This ABB may use other ABBs from the Integration Layer such as the Message Transformer ABB, Auditor ABB, Logger ABB, and Exception Handler ABB. It may leverage the Access Controller ABB from the Quality of Service Layer.

13.2.1.5 Message Transformer

This ABB is responsible for transforming messages from one format to the other, including data format transforms into a single "canonical" form or its subset, and transformation across protocols to whatever protocols that are routed. It also supports data enrichment.

13.2.1.6 Semantic Transformer

This ABB is responsible for semantic/positional mapping of data, so that it conforms to standards. What it does is dependent on the associated standard; e.g., UDEF in general scenarios or in the case of specific lines of business, ICD 10 or LOINC. For example, this ABB transforms and maps a first name and last name in the in-bound data provided by the Mediator ABB to an Integration Layer standard format where the order is reversed. As the semantic interoperability

of services becomes more prevalent with the adoption of cloud computing and SaaS, this becomes more important.

13.2.1.7 Data Transformer

This ABB is responsible for transforming data formats from source to target format; e.g., from proprietary to industry standards and *vice versa*. It integrates with the Information Layer to obtain metadata such as the canonical form, etc.

13.2.1.8 Protocol Converter

This ABB is responsible for transforming data across industry standard protocols. For example, a JSON to SOAP conversion or a SOAP/HTTP to a SOAP/JMS or SOAP/Message Queue conversion would be responsibilities of this ABB.

13.2.1.9 Asynchronous Messaging Manager

This ABB is responsible for storing and forwarding messages potentially using a message queue and it provides the underlying plumbing to transport messages between service invokers and providers as well as the ability to store and forward the messages. It supports both reliable transport and guaranteed delivery. Message queuing is a common example of a mechanism to support asynchronous messaging. A common feature of this ABB is to support message delivery guarantees and reliability.

13.2.1.10 Transaction Manager

This ABB manages transactions and encapsulates transaction handling from the remaining tiers, especially when a composite service invokes a service chain. Types of transaction handling include atomic ACID transactions, two-phase commit, long-running, journaled, and compensating transactions. It leverages standards such as WS* (WS-Coordination, with WS-Atomic Transaction for atomic, ACID transactions and WS-BusinessActivity for long-running transactions) standards to achieve this. Most of the transaction standards specify a set of transaction interaction patterns to address different kinds of transactions.

13.2.1.11 Data Aggregator

This ABB supports the ability to compose (aggregate) data from different services/service providers which are interacting with the Integration Layer using the Mediator ABB into one format, after they have been transformed into a consistent "enterprise" format (canonical form) by the Message Transformer ABB. It is used by the Mediator ABB.

13.2.1.12 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

13.2.1.13 Logger

This ABB is responsible for capturing and recording message routing and service invocation activities. It logs data to monitor system exceptions and system stability (data such as resource availability, etc.). This should be interoperable and integrate with the Quality of Service Layer's monitoring features. It is important from a monitoring and support for compliance perspective.

13.2.1.14 Auditor

This ABB is responsible for tracking and monitoring the message routing and service invocation activities. It supports the capture of audit data, the conversion to standard formats such as XDAS and CBE, the encryption of that data during transport, and the obfuscation of sensitive data.

13.2.1.15 Exception Handler

This ABB is responsible for handling system exceptions raised during service invocation and message passing. System exceptions are caused due to software or hardware errors and not due to application logic errors. Application exceptions are treated as business events and handled through the Event Manager.

13.2.1.16 Event Broker

This ABB is responsible for facilitating event consumers (subscribers, sensors) to subscribe to events and event producers (publishers, emitters) to publish the event and to propagate the event from producers to consumers. It leverages the Asynchronous Messaging Manager ABB, Messaging Transformer ABB, and Router ABB.

13.2.1.17 Event Producer

This ABB is responsible for generating, publishing, emitting, or producing events.

13.2.1.18 Event Listener

This ABB is responsible for registering an interest in a particular type of event; i.e., for subscribing to a particular type of event.

13.2.1.19 Quality of Service Layer: Configuration Manager

See Configuration Manager ABB in the Quality of Service Layer.

13.2.2 Structural Overview of the Layer

The Integration Layer is a critical component of the SOA RA. Its ABBs can be thought of as being logically partitioned into categories which support:

- Ability to provide the integration of services with underlying solutions platforms, to discover services, and, at runtime, to support the virtualization of services so that changes to the end-points (or locations from where the services are called and where the services are provided) can occur without impact to service consumers and service providers
- Ability to support mediation; mediation can be thought of as the routing of the request to correct providers after necessary message transformation and protocol conversion and aggregation of the content from different service providers, from different formats and protocols into a common canonical form (data format) – while not normative, it is customary to support service messages in an XML format after mediation

  — Ability to support different service standards such as WS-Security, etc.

— Ability to mediate reliability through the imposition of WS* and other standards-based protocols

— Ability to support routing and orchestration, including routing based on content (content-based routing), static service composition, and orchestration (calling services in a defined sequence) to deliver data

- Ability to perform message transformation
- Ability to support Quality of Service (QoS) requirements such as transaction management, performance criteria, exception handling, etc.
- Ability to support security requirements
- Ability to track, monitor, and manage service invocations

Figure 41 illustrates the ABBs partitioned into key categories:

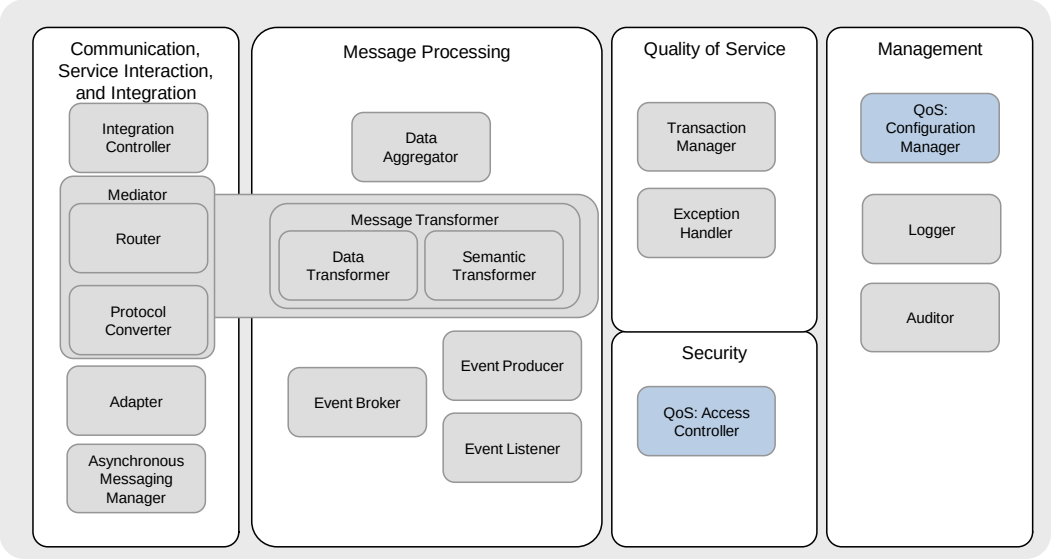


Figure 41: ABBs in the Integration Layer

13.3 Inter-Relationships between the ABBs

The figures below show the inter-relationships and a non-normative sequence of interaction. The final, prescriptive sequence of interaction will be defined by the underlying solution architecture and the standards that the invoking other SOA RA layers support.

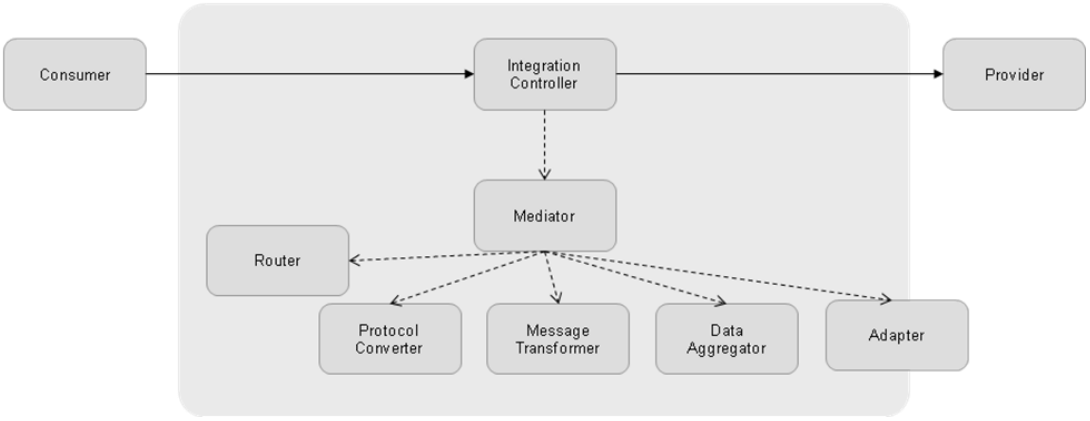


Figure 42: Simple Interactions between Consumer and Provider through the Integration Layer

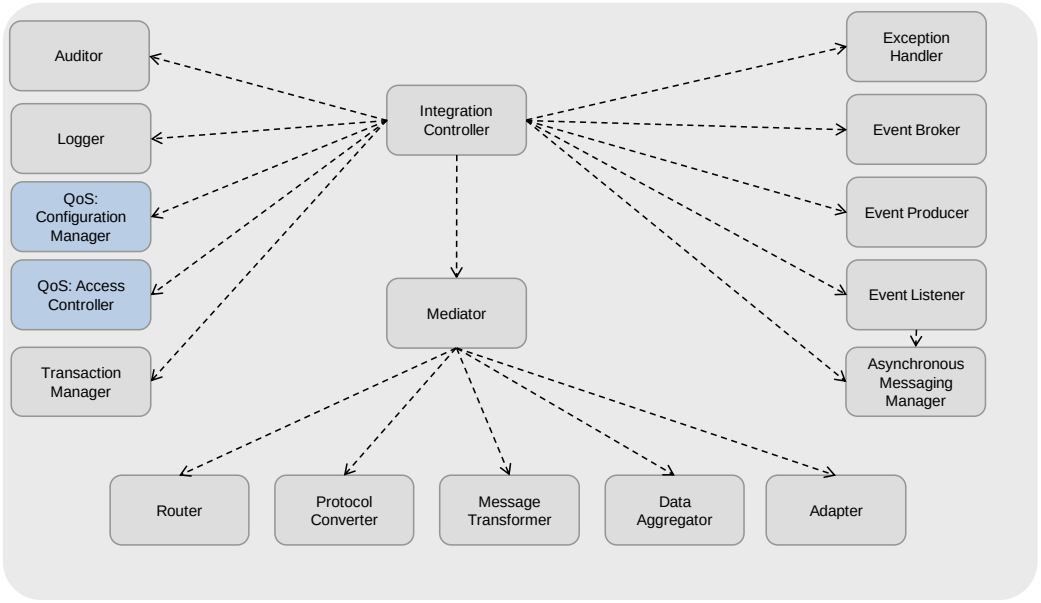


Figure 43: Relationships among ABBs in the Integration Layer

13.4 Significant Intersection Points with other Layers

13.4.1 Interaction with Cross-Cutting Layers

The Integration Layer relies on other cross-cutting layers of the architecture to fulfill its responsibilities.

It relies on the Governance Layer for the following capabilities:

- Ability to store metadata for policies
- Ability to support management (storage, retrieval, etc.) of rules to support rules associated with decision points in service mediation, orchestration, and composition; the Integration

Layer will leverage a common business rules capability that can be used by the ESB (the component in the Integration Layer which mediates – routes and transforms data)

- Ability to determine service end-points for service virtualization

It relies on the Quality of Service Layer for the following capabilities:

- Ability to authenticate/authorize for service invocation and messages

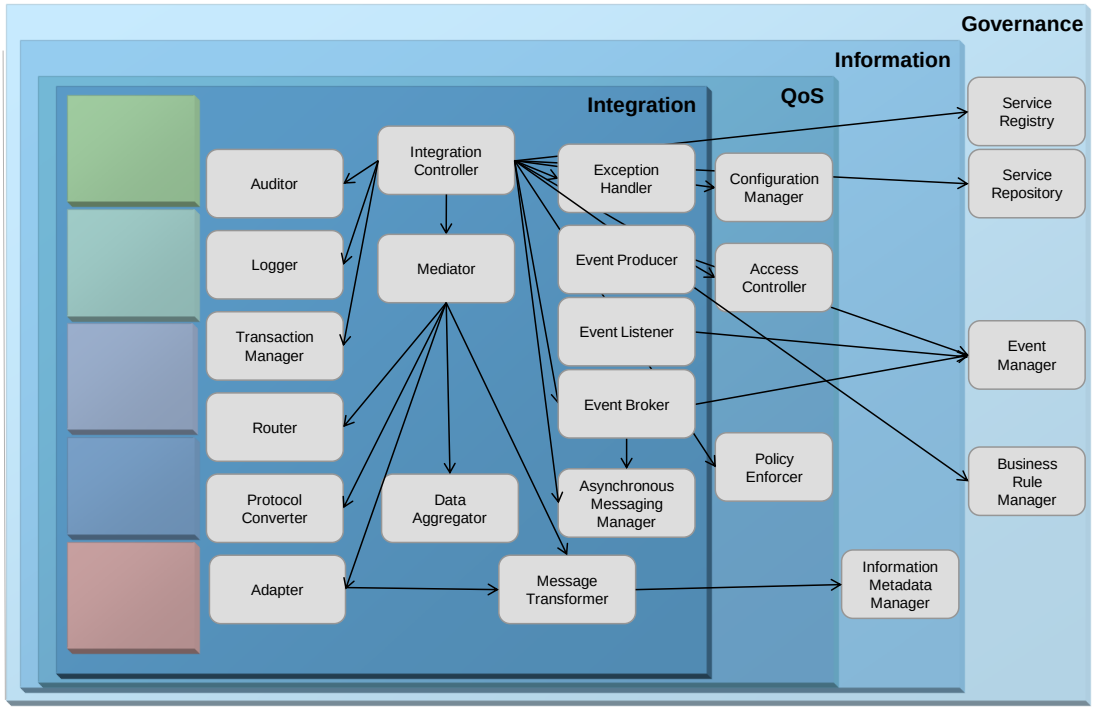


Figure 44: Key Interactions of the Integration Layer with Cross-Cutting Layers

Therefore, the Integration Layer interfaces with the following ABBs of cross-cutting layers of the architecture to provide its capabilities:

- It leverages the Service Registry ABB and Service Repository ABB from the Governance Layer for storing metadata such as policy, schema, etc. and for providing access to the metadata. The Service Registry ABB contains service definitions at runtime and supports service virtualization and service discovery.

- It leverages the Business Rule Manager ABB in the Governance Layer to support rule implementation for the Integration Layer.

- It leverages the Access Controller ABB in the Quality of Service Layer for authenticating/authorizing facility for service invocation and message routing. It also leverages the Policy Enforcer ABB in the Quality of Service Layer to enforce policies local to the Integration Layer.

- The Data Transformer ABB in the Integration Layer uses metadata from the Information Layer and leverages the Information Metadata Manager ABB from the Information Layer for data transformation.
- The Event Broker ABB, Event Listener ABB, and Event Producer ABB in the Integration Layer use the Event Manager ABB in the Governance Layer for event definition and related information.

13.4.2 Interaction with Horizontal Layers

Horizontal layers of the SOA RA leverage ABBs from this layer to provide their respective capabilities. Horizontal layers of the SOA RA use the Integration Controller ABB in the Integration Layer to access the ABBs in the Integration Layer and capabilities they provide, such as Mediator ABB, Router ABB, Message Transformer ABB, Data Transformer ABB, etc.

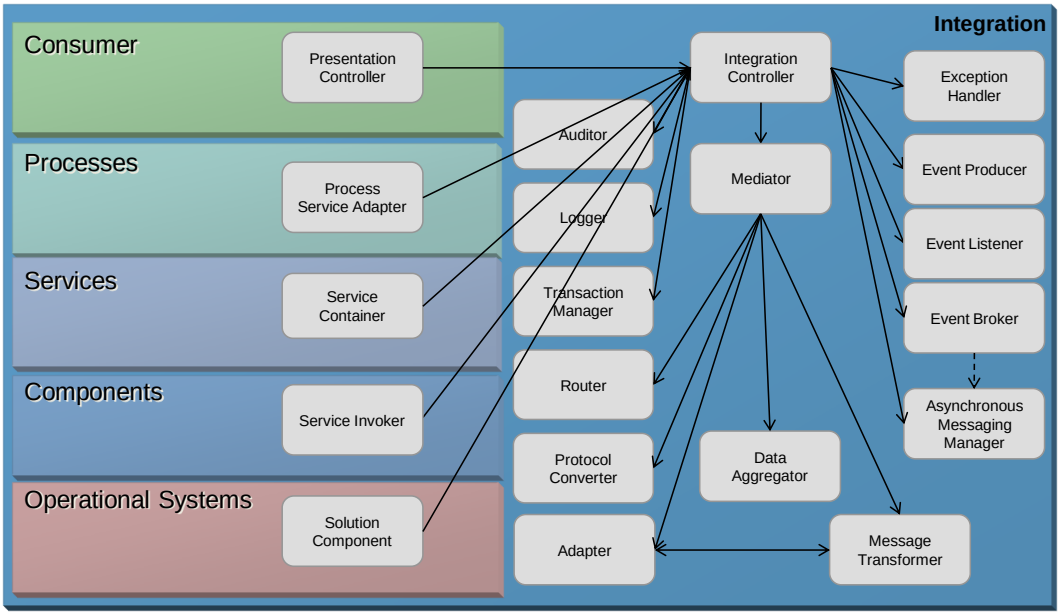


Figure 45: Key Interactions of the Integration Layer with Horizontal Layers

Figure 46 provides a typical interaction with horizontal layers of the SOA RA with the Integration Layer:

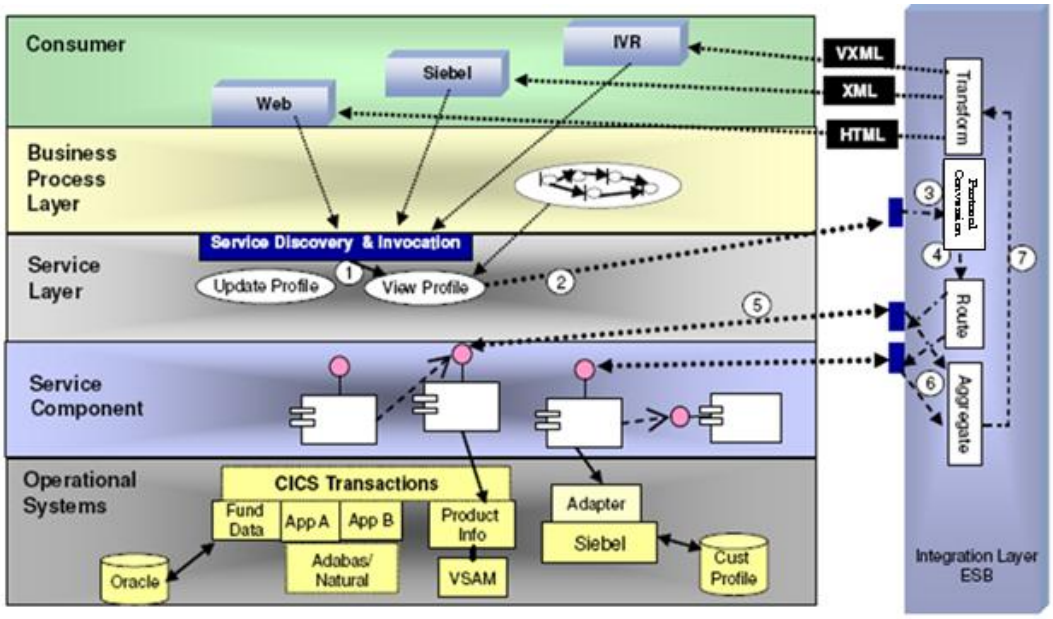


Figure 46: Detail Interactions of the Horizontal Layers with the Integration Layer

In the example above:

- Through the service discovery and invocation process, the web client looks up the View Profile Service.
- A connection is made to the Integration Layer (ESB).
- The ESB performs protocol conversion, if necessary.
- It subsequently routes the call to the appropriate destination.
- It then receives the results of the call.
- It then aggregates the results of the call.
- The aggregated result is transformed and returned to the client in a format that can be consumed by it (for example, in the case of the web client, the aggregated result is returned in an HTML format).

13.5 Usage Implications and Guidance

The Integration Layer also incorporates the support for service virtualization using static routing rules or dynamically at runtime using a service repository and service registry. Service virtualization decouples the location of services for consumers of services. Services are exposed to consumers through a registry, but the exact location decoupled, to support versioning, change of service locality, and administration, without impacting the consumer.

To summarize, the Integration Layer supports capabilities required for enabling SOA such as routing, protocol support and conversion, messaging/interaction style, support for heterogeneous

environment, adapters, service interaction, service enablement, service virtualization, service messaging, message processing, and transformation.

14 Quality of Service Layer

14.1 Overview

Inherent in SOA are characteristics that exacerbate existing Quality of Service (QoS) concerns in computer systems: increased virtualization/loose-coupling, widespread use of XML, the composition of federated services, multiple channels as consumers of services, heterogeneous computing infrastructures, decentralized Service-Level Agreements (SLAs), the need to aggregate IT QoS metrics to produce business metrics, etc. are part of the nature of SOA. These characteristics create complications for QoS that clearly require attention within any SOA solution. The key responsibilities of the Quality of Service Layer include:

- Monitoring and management both at the business level in terms of Key Performance Indicators (KPIs), events, and business activities in the business processes, and at the IT systems level for the security, health, and wellbeing of IT systems, services, applications, networks, storage, and compute servers
- Monitoring and enforcement of a multitude of policies and corresponding business rules including business-level policies, security policies, access privileges, data access policies, etc.

14.1.1 Context and Typical Flow

This layer provides solution QoS management of various aspects, such as availability, reliability, security, and safety as well as mechanisms to support, track, monitor, and manage solution QoS control.

The Quality of Service Layer provides the service and SOA solution lifecycle processes with the capabilities required to ensure that the defined policies, Non-Functional Requirements (NFRs), and governance regimens are adhered to.

This layer supports monitoring and capturing service and solution metrics in an operational sense and signaling non-compliance with NFRs relating to the salient service qualities and policies associated with each SOA layer. Service metrics are captured and connected with individual services to allow service consumers to evaluate service performance, creating increased service trust levels.

This layer serves as an observer of the other layers and can create events signaling when a non-compliance condition with salient policies is detected or (preferably) when a non-compliance condition is anticipated.

In the SOA RA policies, business rules, and the NFRs and policies for the SOA solution are defined and captured in the Governance Layer but are monitored and enforced in the Quality of Service Layer. Important areas of policy enforcement are security, messaging transportation, and infrastructure availability, and service availability. This layer also supports security management

and systems management for SOA solutions. Responses (dispensations and appeals) to non-compliance and exceptions are defined by the Governance Layer as well.

14.1.2 Capabilities

There are multiple sets of categories of capabilities that the Quality of Service Layer needs to support in the SOA RA. These categories are:

- **Command and Control Management**: This category of capabilities provides the command center for security management as well as the operational security capabilities for non-IT assets and services to ensure protection, response, continuity, and recovery. It also supports security of physical assets such as locations, facilities, services, inventory, physical access control, human identity, etc.
- **Security Management**: This category of capabilities manages and monitors security and secure solutions. This provides the ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, address how software, systems, and services are developed and maintained throughout the software lifecycle, maintain the security status through proactive changes reacting to identified vulnerabilities and new threats, enable the IT organization to manage IT-related risks and compliance, and provide the automation basis for security management.
- **IT Systems Monitoring and Management**: This category of capabilities provides monitoring and management of IT infrastructure and systems. This includes the ability to monitor and capture metrics and status of IT systems and infrastructure.
- **Application and SOA Monitoring and Management**: This category of capabilities provides monitoring and management of software services and applications. This includes the ability to capture metrics and to monitor and manage application and solution status.
- **Business Activity Monitoring and Management**: This category of capabilities provides monitoring and management of business activities and business processes. It provides the ability to analyze this event information, both in real-time/near real-time, as well as stored (warehoused) events, and to review and assess business activities in the form of event information and determine responses or issues alerts/notifications.
- **Event Management**: This category of capabilities provides the ability to manage events and enables the complex event processing in the SOA RA.
- **Policy Monitoring and Enforcement**: This category of capabilities provides a mechanism to monitor and enforce a multitude of policies and corresponding business rules including business-level policies, security policies, access privileges, and data access policies. This provides the ability to find and access policies, evaluate and enforce policies at checkpoints or on metrics captured, signal and record compliance status or metrics, send notification and log of non-compliance and change rules, policies, configuration, and status.
- **Configuration and Change Management**: This category of capabilities provides the ability to change solution configuration and descriptions.

- **Data Repository**: This category of capabilities provides the ability to store and access policies and rules.

This layer features the following capabilities:

Command and Control Management

1. Ability to ensure protection, response, continuity, and recovery
2. Ability to approve authority for security
3. Ability to ensure that physical and operational security is maintained for locations, assets, humans, environment, and utilities
4. Ability to provide surveillance and monitoring of locations, perimeters, and areas
5. Ability to enforce entry control
6. Ability to provide for positioning, tracking, and identification of humans and assets; continuity, and recovery operations
7. Ability to secure physical assets, such as locations, facilities, services, inventory, physical access control, human identity, etc.
8. Ability to ensure the safety of a solution from types of failure, damage, error, accidents, and harm as defined by the Governance Layer

Security Management

9. Ability to ensure appropriate authentication based on proper roles
10. Ability to ensure appropriate authorization based on proper roles
11. Ability to ensure appropriate encryption of messages
12. Ability to ensure appropriate audit logging of messages
13. Ability to assure that access to resources has been given to the right identities, at the right time, for the right purpose
14. Ability to monitor and audit access to resources for unauthorized or unacceptable use
15. Ability to protect unstructured and structured data from unauthorized access and data loss, according to the nature and business value of information
16. Ability to monitor and audit access to information
17. Ability to address how software, systems, and services are designed, developed, tested, operated, and maintained throughout the software lifecycle including the use of technology as well as processes and procedures which are followed during all aspects of software development and deployment

18. Ability to maintain the security status through proactive changes to the system, reacting to identified vulnerabilities and new threats, and through responding to detected incidents and reported problems
19. Ability to identify, quantify, assess, and report on IT-related risks that contribute to the enterprise's operational risk by providing all services to analyze and report security information and security events, and create alarms and insight
20. Ability to provide the automation basis for security management
21. Ability to provide and enforce policies for access control
22. Ability to control access to individual data items in messages

IT Systems Monitoring and Management

23. Ability to monitor, manage, and configure IT systems hardware, including operating systems which are part of an SOA solution
24. Ability to monitor, manage, and configure IT network hardware systems which are part of an SOA solution
25. Ability to monitor, manage, and configure IT storage hardware systems which are part of an SOA solution

Application and SOA Monitoring and Management

26. Ability to coordinate overall QoS requirements for the SOA solution
27. Ability to describe QoS NFRs
28. Ability to manage solutions and services from solution delivery to solution termination
29. Ability capture the percentage of executions that the solution does not fail
30. Ability to capture the percentage of executions of the solution that execute within a prescribed period of time
31. Ability to capture the metric for percentage of time that the solution is invokable
32. Ability to capture the metric for the response time of network access to a service or solution
33. Ability to react to infrastructure changes to maximize availability
34. Ability to log or report on availability metrics
35. Ability to evaluate the availability metrics against the NFRs (policy)
36. Ability to capture metrics on performance of services and solutions
37. Ability to change configuration and policy to ensure meeting SLAs
38. Ability to change configuration and policy to ensure performance optimization

39. Ability to support virtualization of resources to support performance optimization
40. Ability to record, track, and monitor the cost of executing a specific solution
41. Ability to monitor current status of the solution
42. Ability to change the current status of the solution
43. Ability to check QoS requirements for valid status
44. Ability to issue events for non-compliance to QoS requirements
45. Ability to measure, gather, evaluate, and test metrics against policies on a regular basis

Business Activity Monitoring and Management

46. Ability to analyze this event information, both in real-time/near real-time, as well as stored (warehoused) events
47. Ability to review and assess business and service activity in the form of event information and determine responses or issue alerts/notifications

Event Management

48. Ability to interface with the Integration Layer and obtain events from the Integration Layer
49. Ability to control issuance of events in the solution
50. Ability to send issue events indicating non-compliance to QoS requirements
51. Ability to subscribe to events issued by the solution
52. Ability to log events and business messages
53. Ability to control logging frequency and size

Policy Monitoring and Enforcement

54. Ability to check QoS requirements for valid rules
55. Ability to change rules to comply with QoS requirements
56. Ability to change QoS requirements to comply with rules
57. Ability to send events for non-compliance to QoS requirements
58. Ability to evaluate policies
59. Ability to resolve conflicts between policies
60. Ability to enforce compliance with policies
61. Ability to automatically respond and correct violations of policies (enforce)
62. Ability to enable policy enforcement

63. Ability to discover, analyze, transform, distribute, evaluate, and enforce security policies
64. Ability to manage non-functional QoS solution requirements from solution delivery to solution termination
65. Ability to manage lifecycle of policies
66. Ability to represent policies
67. Ability to author policies
68. Ability to manage instances of policies
69. Ability to change policies
70. Ability to disable, discard, and discontinue policies
71. Ability to monitor and capture metrics and status
72. Ability to find and access policies
73. Ability to evaluate policies at checkpoints or on metrics captured
74. Ability to automate monitoring for violations of policy

Configuration and Change Management

75. Ability to capture configuration (authoring tools)
76. Ability to change configuration
77. Ability to check QoS requirements for valid configurations
78. Ability to change configuration to comply with QoS requirements
79. Ability to send events for non-compliance to QoS requirements
80. Ability to track and record changes, configuration, metadata, policy, etc., happening in the solution
81. Ability to recover from or even reverse changes made to the solution
82. Ability to ensure that changes are executed in compliance with relevant governance policies
83. Ability to change metadata, including service descriptions
84. Ability to propagate metadata changes to other repositories and descriptions

Data Repository

85. Ability to store QoS policies and rules
86. Ability to locate/find/return QoS policies and rules

14.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Quality of Service Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Command and Control Management	Command and Control Manager	1-6
2		Security-Aware Physical Asset Manager	7
3		Safety Manager	8
4	Security Management	Security Manager	9-12
5		Identity, Access, and Entitlement Manager	13-14
6		Data and Information Protector	15-16
7		Software, System, and Service Assurer	17
8		Threat and Vulnerability Manager	18
9		Risk and Compliance Assessor	19
10		Security-Aware Service Manager	20
11		Access Controller	21
12		Data-Driven Access Controller	22
13	IT Systems Monitoring and Management	IT Systems Manager	23-25
14		Server and Systems Manager	23
15		Network Manager	24
16		Storage Manager	25
17	Application and SOA Monitoring and Management	Services and Application Manager	26-45
18		Solution Manager	26 -27
19		Status Manager	41-44
20		Lifecycle Manager	28
21		Reliability Manager	29-30
22		Availability Manager	31-35
23		Performance Manager	36-39
24		Execution Cost Manager	40

#	Capability Category	ABB Name	Supported Capabilities
25		Monitoring Metric Tools	45
26	Business Activity Monitoring and Management	Business Activity Manager	46-47
27		Business Activity Monitor	46-47
28		Activity Correlation Manager	46-47
29	Event Management	Event Manager	48-51
30		Integration Layer: Event Producer	50, 57
31		Integration Layer: Event Listener	51
32		Logging Manager	52-53
33	Policy Monitoring and Enforcement	Policy Enforcer	58-62, 70
34		Policy Monitor	71-74
35		Governance Layer: Policy Manager	63-70
36		Governance Layer: Business Rules Manager	54-56
37	Configuration and Change Management	Configuration Manager	75-79
38		Metadata Manager	73-84
39		Governance: Change Control Management	80-82
40	Data Repository	Governance: Repository	85-86

Table 7: ABB to Capability Mapping for the Quality of Service Layer

14.2 Details of ABBs and Supported Capabilities

14.2.1 Details of ABBs

This section describes in detail each ABB in terms of its responsibilities.

14.2.1.1 Command and Control Manager

This ABB provides the command center for security management as well as the operational security capabilities for non-IT assets and services to ensure protection, response, continuity, and recovery. Its responsibilities include:

- Providing the approving authority for security

- Ensuring that physical and operational security is maintained for locations, assets, humans, environment, and utilities
- Providing surveillance and monitoring of locations, perimeters, and areas
- Enforcing entry controls
- Providing for positioning, tracking, and identification of humans and assets; continuity and recovery operations

14.2.1.2 Security-Aware Physical Asset Manager

This ABB supports the ability to secure physical assets such as locations, facilities, services, inventory, physical access control, human identity, etc. Its responsibilities include physical and electronic surveillance, location perimeter control, asset and human identification, and tracking.

14.2.1.3 Safety Manager

This ABB is responsible for handling the safety features of a solution. A solution is considered safe if it is protected against predefined types of failure, damage, error, accidents, and harm.

14.2.1.4 Security Manager

This ABB is responsible for handling the security features of a solution. A solution is considered to have high security if it ensures authentication and authorization based upon proper roles.This ABB also changes, configures, and audits the security of the compliance, dispensation, and communication processes for the Governance Layer. It provides the binding to whatever standards policies are defined in the Governance Layer and the ability to enforce them (acting as a Policy Enforcer for security policies).

14.2.1.5 Identity, Access, and Entitlement Manager

This ABB is responsible for all capabilities related to roles and identities, access rights, and entitlements. This ABB provides trust management, identity lifecycle management, credential management, role entitlement, and compliance management. The goal of this ABB is to assure that access to resources has been given to the right identities, at the right time, for the right purpose. It also supports that access to resources is monitored and audited for unauthorized or unacceptable use. It also provides the ability to monitor and audit access to resources for unauthorized or unacceptable use.

14.2.1.6 Data and Information Protector

This ABB protects unstructured and structured data from unauthorized access and data loss while providing according to the nature and business value of information. It also ensures that access to information is monitored and audited.

14.2.1.7 Software, System, and Service Assurer

This ABB addresses how software, systems, and services are designed, developed, tested, operated, and maintained throughout the software lifecycle including the use of technology as well as processes and procedures which are followed during all aspects of software development and deployment. Its responsibilities include:

- Structured design process
- Threat modeling
- Risk assessment
- Design reviews for security
- Source code review
- Source code analysis
- Dynamic application analysis
- Source code control
- Access monitoring
- Code/package signing and verification
- Quality assurance testing
- Supplier and third-party code validation
- Security problem and incident management of software and services

14.2.1.8 Threat and Vulnerability Manager

This ABB maintains the security status through proactive changes to the system, reacting to identified vulnerabilities and new threats, and through responding to detected incidents and reported problems. Its responsibilities includes vulnerability testing, vulnerability scanning, virtual patching, threat analysis, and risk analysis.

14.2.1.9 Risk and Compliance Assessor

This ABB enables IT organizations to identify, quantify, assess, and report on IT-related risks that contribute to the enterprise's operational risk by providing all services to analyze and report security information and security events, and create alarms and insight. Its responsibilites include: security and compliance dashboard, forensics, reporting specially risk aggregration and reporting, compliance audit.

14.2.1.10 Security-Aware Service Manager

This ABB provides an automation basis for security management, including tie-backs to service management disciplines such as incident and problem management, change and release management, and asset management.

14.2.1.11 Access Controller

This ABB acts as a kind of policy enforcer providing access control and enforcing policies related to access control and rights. This includes the enforcement of "trust" policies such as authentication/authorization facilities for service invocation and message routing as well as access privileges for various participants to data. It typically supports authorization and authentication functionalities for registered participants, including federated authentication

(single sign-on) and the ability to ensure that the appropriate audit logging is carried out. This ABB depends on the Governance Layer, that defines security policies, to retrieve security policies and act as a local policy decision point and local Policy Enforcement Point (PEP). It can include support for standards such as SAML (authentication and authorization), XDAS, and CBE (audit and logging). It leverages the Identity, Access, and Entitlement Manager ABB to fulfill its responsibilities.

14.2.1.12 Data-Driven Access Controller

This ABB supports access control on individual data items. It is a specialized kind of access controller and policy enforcer that enforces policies on individual data items. For example, in a claim processing scenario by an insurance provider, the tax identification number of a claimant is only to be viewed by a set of individuals who are certified to handle sensitive personal information. It leverages the Data and Information Protector ABB to fulfill its responsibilities.

14.2.1.13 IT Systems Manager

This ABB is responsible for the coordination of the systems manager, network manager, and storage manager to manage the SOA solution elements in a runtime environment.

14.2.1.14 Server and Systems Manager

This ABB is responsible for the systems manager of the runtime environment.

14.2.1.15 Network Manager

This ABB is responsible for monitoring network infrastructure performance, proactively identifying potential network issues and problems, and isolating and correcting network faults.

14.2.1.16 Storage Manager

This ABB is responsible for managing storage resources, including access to local, networked, and virtualized storage.

14.2.1.17 Services and Application Manager

This ABB is responsible for monitoring and managing the overall health of the applications such that an application must be available to be used (availability), must perform reasonably as stated in the NFRs (performance), must handle the information correctly (integrity), and must be able to recover the data that it has (reliability). Integrity and reliability are typically handled inside the application which uses several redundant storage and commit mechanisms to achieve integrity and reliability. On the other hand, the availability and performance of the application depends on its components that support the application and relationship and interconnection among the components. This ABB is responsible for understanding these relationships and presents the root cause of the application problem. This includes decomposing the application and understanding the individual component resource needs to be able to pinpoint resource problems on an application context. This ABB includes data collection agents responsible for collecting data and monitoring information in application servers. These collection agents run in the monitored application servers and send monitoring information to the management server. There could be specific collection agents for different types of environment such application servers, mainframe/CICS, etc. This ABB also includes a management server that serves as heart and

brain of this ABB and is responsible for processing the data collected and sent by data collection agents and presents management data on a dashboard.

14.2.1.18 Solution Manager

This ABB is the center of the Quality of Service Layer. It coordinates the management of the solution and all of the other ABBs. This ABB is responsible for coordinating solution lifecycle, security, availability, configuration, and change.

14.2.1.19 Status Manager

This ABB supports the ability to track and change the lifecycle and availability status of services. It is used by the Services Layer.

14.2.1.20 Lifecycle Manager

This ABB is responsible for managing solution-level QoS requirements during the period of a solution's lifecycle, from the time when the solution is delivered to the time when the solution is terminated or discarded.

14.2.1.21 Reliability Manager

This ABB is responsible for handling the reliability feature of a solution. It refers to the percentage that a solution can be successfully executed without failure during a certain period of time.

14.2.1.22 Availability Manager

This ABB is responsible for handling the availability feature of a solution. Since a solution here refers to an SOA-oriented business solution, it implies a network-based service accessible remotely. Due to unpredictable network features, a solution is considered with high availability if its delay is always below some predefined threshold.

14.2.1.23 Performance Manager

This ABB is responsible for capturing metrics on performance of services and solutions and recording or reporting these metrics if they do not adhere to relevant policies or they exceed thresholds. This ABB can change configuration and policy to ensure meeting SLAs and/or to ensure performance optimization. This ABB may be expected to support management of virtualized resources in order achieve performance optimization.

14.2.1.24 Execution Cost Manager

This ABB is responsible for recording, tracking, and monitoring the cost needed to execute a specific solution.

14.2.1.25 Monitoring Metric Tools

This ABB measures, gathers, evaluates, and tests metrics against policies on a regular basis. Metrics are gathered on SOA services, governed processes, and governing processes. This ABB interacts with the Policy Enforcer ABB.

14.2.1.26 Business Activity Manager

This ABB enables the event information to be analyzed, both in real-time/near real-time, as well as stored (warehoused) events using the Activity Correlation Manager ABB. It provides event-based analytic functionality, the ability to perform scenario analysis, and sense and respond capability. It uses the Activity Correlation Manager ABB to carry out complex, real-time/near real-time analysis to determine and trigger complex events as well as render real-time trends in business activity. It uses different channels to support alerts and notification about the occurrence of events, and supports continuous monitoring of events. This capability helps organizations to proactively react to both threats as well as opportunities. An example of an opportunity might be a customer's pattern of buying triggering a sales recommendation or particular business process flow. An example of threats might be the loading of processes by a particular key insurance quote process, with the trending to failure, or the occurrence of a particular sequence of events in a nuclear power plant.

14.2.1.27 Business Activity Monitor

This ABB monitors the event, business activities in a business processes, and services. It interfaces with the Integration Layer to handle notification and propogation of events.

14.2.1.28 Activity Correlation Manager

This ABB reviews and assess inbound business and service activity in the form of event information and determines responses or issues alerts/notifications.

14.2.1.29 Event Manager

This ABB controls the issuance of events in the solution. It controls the ability to issue and subscribe to events and any logging or processing of the events.

14.2.1.30 Integration Layer: Event Producer

See Event Producer ABB in the Integration Layer.

14.2.1.31 Integration Layer: Event Listener

See Event Listener ABB in the Integration Layer.

14.2.1.32 Logging Manager

This ABB is responsible for configuring and enabling logging of events and business messages. Logging frequency and log size should be configurable.

14.2.1.33 Policy Enforcer

This ABB is responsible for enforcing QoS policies including security policies and serves as Policy Enforcement Points (PEPs) in all the horizontal layers and other cross-cutting layers in the SOA RA. It is important to note that this ABB logically also includes the responsibilities of a security policy enforcer.

This ABB interacts with the Governance Layer to retrieve the policies stored there and enforce them locally in each layer. It provides the binding from whatever standards or formats the

policies are written in to the formats needed to enable the ability to enforce them. The intersection or instantiation of this ABB for different layers of the SOA RA represent the PEPs in the architecture.

This ABB may execute diverse policies associated with lower-level message functionality such as addressing transformation, routing, caching, compression, and other content handling functions provided by Integration Layer and other layers in the SOA RA. The specific packaging functionality varies among vendor implementations and can be packaged in software and/or hardware (firmware).

The places where policies are actually applied and enforced change depending on the lifecycle stage.

The registry/repository itself is the point of enforcement during design time. Policies are generally enforced by the underlying message transport system that connects service providers with consumers during runtime. Often, the service container provides the ability to enforce policies, ensuring the services' compliance with their SLAs. Finally, policies are typically enforced by the IT management system during maintenance and manage phase of the lifecycle.

This ABB representing the PEPs can be implemented in software or as a standalone network device such as an appliance. It is any hardware-based, high-performance functional component that intercepts, inspects, filters, and performs content-aware policy-driven processing on application messages and their payloads. It can also be implemented in conjunction with other network device functionality: co-processors, proxies, gateways, blade servers, routers, grids, and other configurations.

The Policy Manager ABB in the Governance Layer sets and updates the policies to be enforced. The Policy Monitor ABB monitors to ensure the policy enforcers are executing correctly.

14.2.1.34 Policy Monitor

This ABB enables automation of monitoring for violations of policy. It includes checkpoints in SOA processes and is an integral part of compliance processes policy. This ABB obtains its policies from the Policy Manager ABB in the Governance Layer. This ABB is passive and interacts with the Policy Enforcer ABB to take any actions when violations are detected. It is responsible for:

- Capturing real-time collection and statistical analysis for display
- Providing a management console for visibility into the management of distributed network of PEPs and the status of these enforcements
- Logging and aggregating measurements and highlighting significant events
- Correlating, analyzing, and visualization of data fed in by the Policy Enforcer ABB at various PEPs

14.2.1.35 Governance Layer: Policy Manager

See Policy Manager ABB in the Governance Layer.

14.2.1.36 Governance Layer: Business Rules Manager

See Business Rules Manager ABB in the Governance Layer.

14.2.1.37 Configuration Manager

This ABB is a set of tools used to define the configuration of SOA solution and processes being governed, as well as to configure tools used to implement and enforce governance. These tools may be driven in an automated way to adjust configurations based on monitoring, policy enforcement, compliance, and dispensation processes.

Ideally, it also supports identifying and preventing improper configurations based on dependencies between ABBs. It enables dynamic configuration of ABBs on-demand. If ABBs are fine-grained, they will be more flexible if they are configured based on specified rules. This configuration can be handled in the following two ways:

- Through template-based configuration, where a user can select a specific template based on the corresponding service request scenario. The system will select all the rules associated with this template and configure the ABBs to support the rules. This requires scenario templates to be created and stored in a repository to be selected when needed.
- Through dynamic template creation, where a user selects certain characteristics and the system will determine the appropriate rules and configure using relevant ABBs at runtime. For instance, a user may require that the system adopt an industry messaging standard and satisfy some SLAs. Based on these requirements the system during runtime will select the appropriate data transformation, protocol conversion, and service providers that meet the SLAs.

14.2.1.38 Metadata Manager

This ABB is responsible for managing metadata in repositories.

14.2.1.39 Governance: Change Control Manager

See Change Control Manager ABB in the Governance Layer.

14.2.1.40 Governance: Repository

See Repository ABB in the Governance Layer.

14.2.2 Structural Overview of the Layer

The ABBs in the Quality of Service Layer can be thought of as being logically partitioned into categories which support:

- Ability to provide the command center for security management as well as the operational security capabilities for non-IT assets and services
- Ability to manage and monitor security and secure solutions
- Ability to monitor and manage IT infrastructure and systems
- Ability to monitor and manage software services and applications

- Ability to monitor and manage business activities and business processes and associated KPIs
- Ability to manage events
- Ability to monitor and enforce a multitude of policies and corresponding business rules
- Ability to change solution configuration and descriptions
- Ability to store and access policies and business rules

Figure 47 illustrates the ABBs partitioned into key categories:

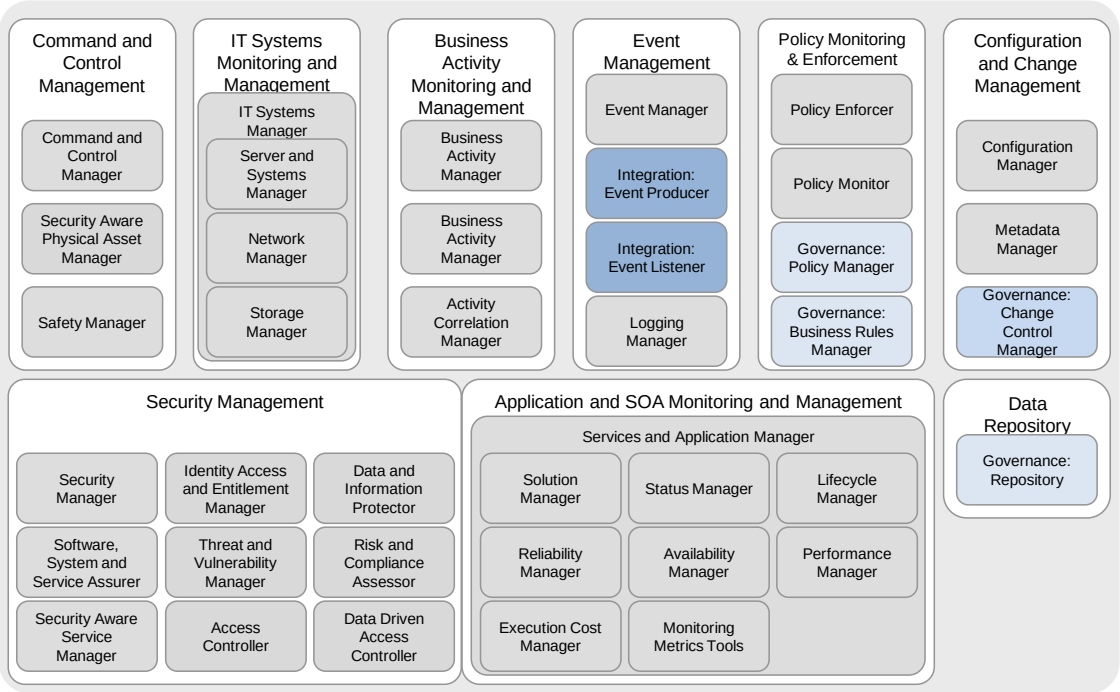


Figure 47: ABBs in the Quality of Service Layer ABB

14.3 Inter-Relationships between the ABBs

Figure 48 illustrates the key relationships amongs the ABBs in the Quality of Service Layer for management of the solution in an operational environment.

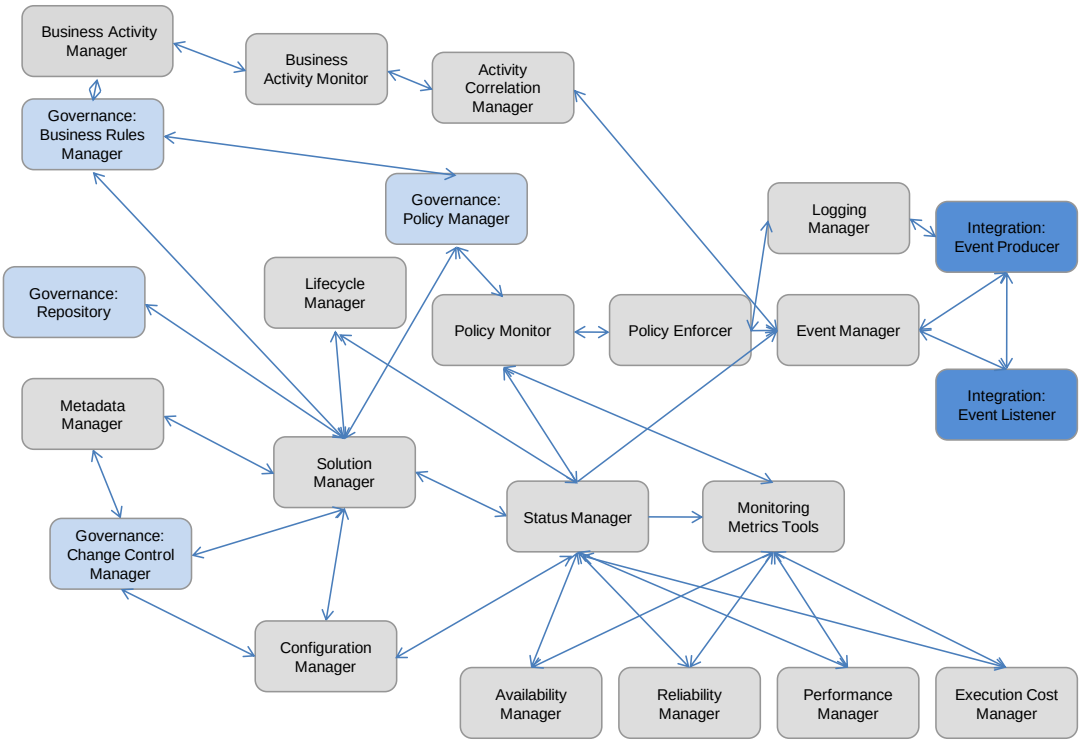


Figure 48: Relationships among ABBs in the Quality of Service Layer

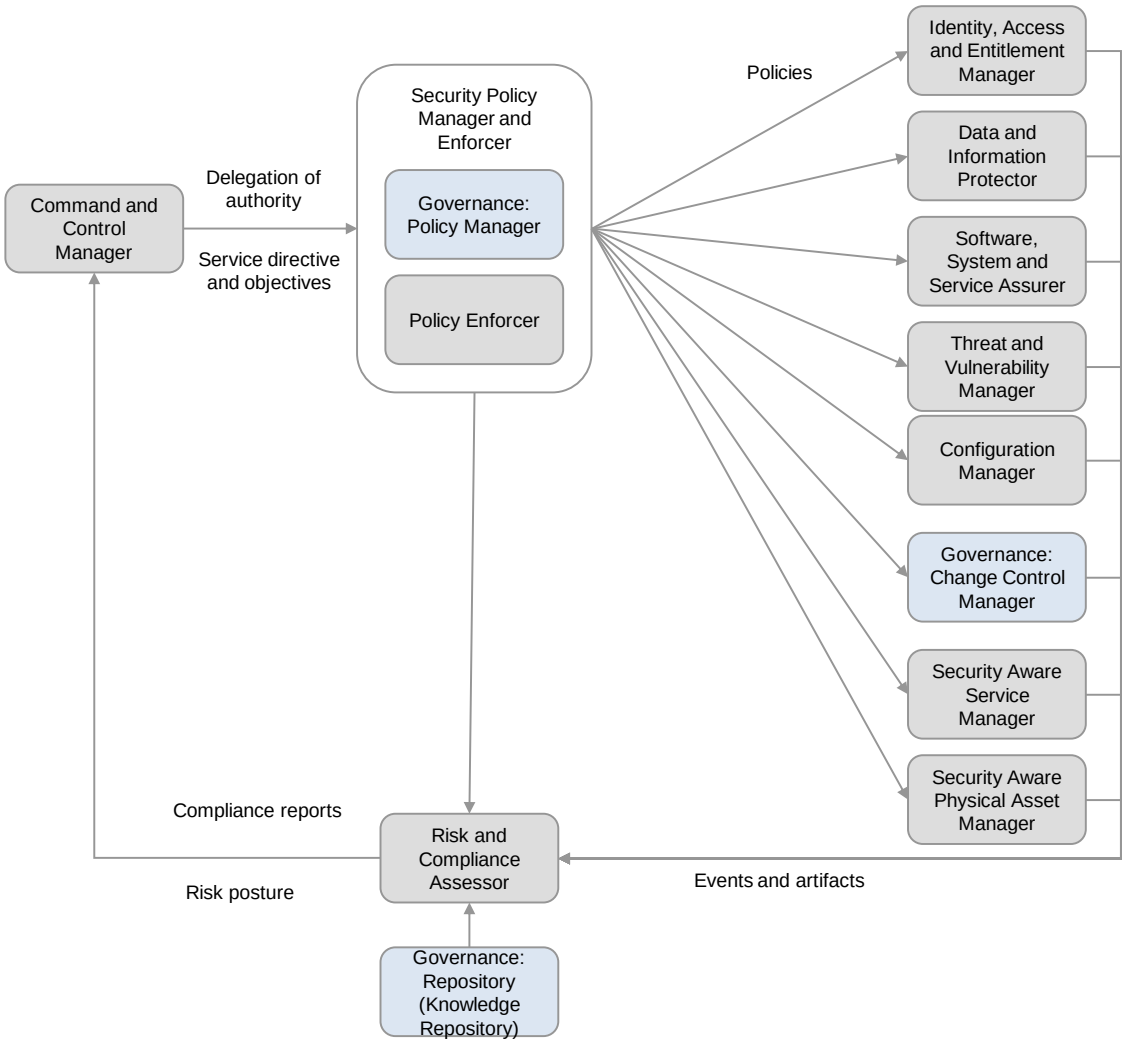


Figure 49: Relationships among ABBs for Command and Control Management and Security Management in the Quality of Service Layer

14.4 Significant Intersection Points with other Layers

14.4.1 Interaction with Cross-Cutting Layers

The Quality of Service Layer depends on other cross-cutting layers in the SOA RA to fulfill its responsibilities:

- It depends on Integration Layer for service integration (adapters), service mediation, message routing and transport, asynchronous messaging, event brokering and listening, transaction management, data aggregation, message, semantic, data and protocol transformation, and exception handling. The Solution Manager ABB uses the Event Producer ABB and Event Listener ABB in the Integration Layer to produce and listen to events.

- It depends on the Governance Layer for definition of policies and associated business rules and responses (dispensations and appeals) to non-compliance and exceptions. The Solution Manager ABB works with the Repository ABB and Policy Manager ABB in the Governance Layer. The relationship of the Quality of Service Layer with the Governance Layer is significant because the Governance Layer contains the processes for identifying and setting the business policies and objectives that generate the QoS NFRs.
- It depends on the Information Layer for definition of events.

The Quality of Service Layer is used by other cross-cutting layers to fulfill their respective responsibilities; for example:

- The Policy Enforcer ABB is leveraged by the Integration, Information Architecture, and Governance Layers to enforce multitude of policies for the respective layers.
- The Access Controller ABB is leveraged by the Integration, Information Architecture, and Governance Layers to enforce security and access control policies for the respective layers.
- The Data-Driven Access Controller ABB is leveraged by Information Layer to enforce access control policies based on individual data items.

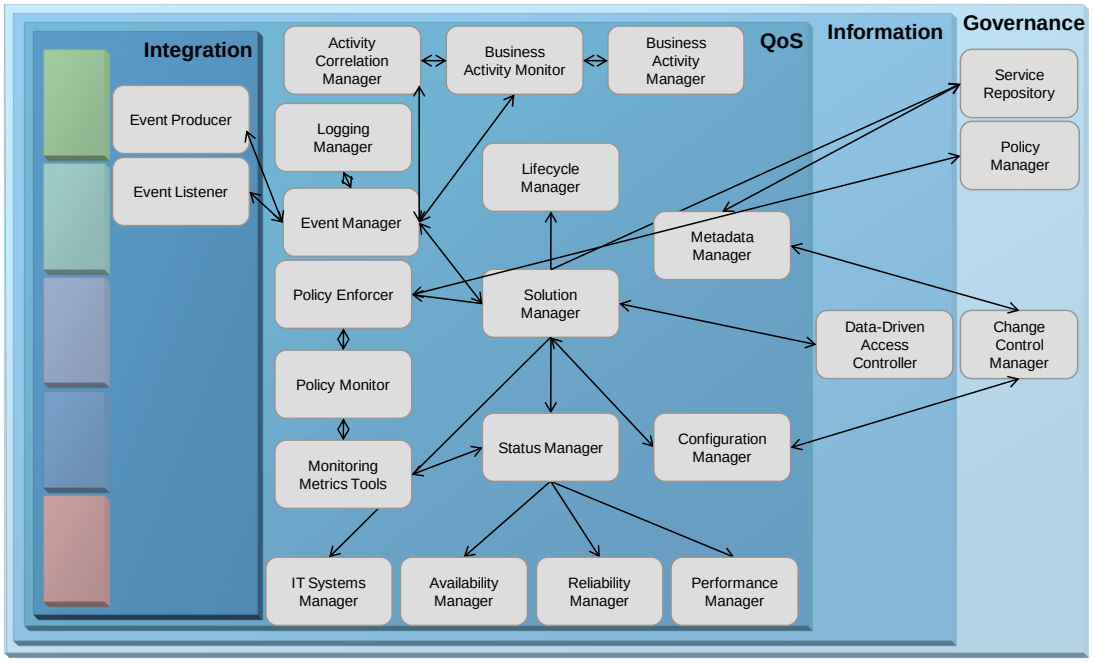


Figure 50: Key Interactions of the Quality of Service Layer with Cross-Cutting Layers

14.4.2 Interaction with Horizontal Layers

It should be noted that the Solution Manager ABB interacts with all other layers: Consumer Layer, Business Process Layer, Service Layer, Service Component Layer, Operational Systems Layer, Integration Layer, Information Layer, and Governance Layer. There are other specific uses of the Quality of Service Layer and its ABBs by horizontal layers such as:

- The Policy Enforcer ABB is leveraged by the Consumer, Business Process, Service, and Service Component Layers to enforce a multitude of policies for the respective layers.
- The Access Controller ABB is leveraged by the Consumer, Business Process, Service, and Service Component Layers to enforce security and access control policies for the respective layers.
- The IT Systems Manager ABB manages all the resources in the Operational Systems Layer.
- The Status Manager ABB is updated by the Service Container when a service changes status.
- All layers collaborate with the Quality of Service Layer via the Solution Manager ABB which coordinates the QoS and security needs of the SOA solution.

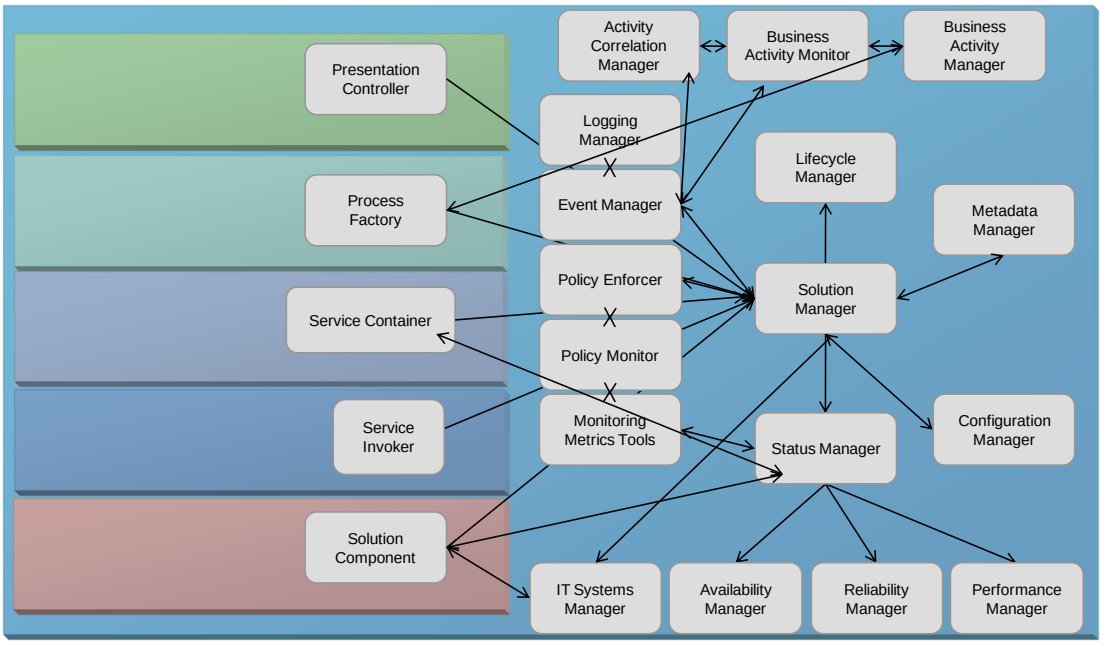


Figure 51: Key Interactions of the Quality of Service Layer with Horizontal Layers

14.5 Usage Implications and Guidance

The Quality of Service Layer establishes NFR-related issues as a primary feature/concern of SOA and provides a focal point for dealing with them in any given solution. It provides the means of ensuring that an SOA meets its requirements with respect to, for example:

- Reliability
- Availability
- Manageability
- Scalability

- Security

Finally, it enhances the business value of SOA by enabling businesses to monitor the business processes contained in the SOA with respect to the business KPIs that they influence.

A significant issue with SOA is security due to its potential perimeter-less nature as opposed to the traditional, web-based, "within the firewall" kind of application. SOA security, which is perimeter-based security, is a capability realized by the Quality of Service Layer.

15 Information Layer

15.1 Overview

15.1.1 Context and Typical Flow

The Information Layer is responsible for manifesting a unified representation of the information aspect of an organization as provided by its IT services, applications, and systems enabling business needs and processes and aligned with the business vocabulary – glossary and terms. Associated with the primary objective of this layer are a number of capabilities. This layer includes information architecture, business analytics and intelligence, metadata considerations, and ensures the inclusion of key considerations pertaining to information architectures that can also be used as the basis for the creation of business analytics and business intelligence through data marts and data warehouses. This includes metadata content that is stored in this layer. It also supports the ability for an information services capability, enabling a virtualized information data layer capability. This enables the SOA to support data consistency, and consistency in data quality.

In particular, this layer can be thought of as supporting multiple categories of capabilities of the SOA RA:

- Ability to support information services capability, critical to support a shared, common and consistent expression of data
- Ability to integrate information across the enterprise in order to enable information services capability
- Ability to define metadata that is used across the SOA RA and in particular the metadata that is shared across the layers
- Ability to secure and protect information
- Ability to support business activity monitoring and critical to the usage of the SOA RA and its realization

In particular, an information virtualization and information service capability typically involves the ability to retrieve data from different sources, transform it into a common format, and expose it to consumers using different protocols and formats.

15.1.2 Capabilities

There are multiple set of categories of capabilities that the Information Layer needs to support in the SOA RA. These categories are:

- **Information Services**: This category of capabilities addresses the support of information services. Information services provide a uniform way of representing, accessing,

maintaining, managing, analyzing, and integrating data and content across heterogeneous information sources. There are primarily two approaches to achieving that. First approach focuses on building a single view of business-critical data for customers, products, location, and others delivered in context; i.e., single view of enterprise (MDM) approach. The second approach focuses on integrating the appropriate information in a timely and consistent manner, analyzing and attempting to improve the quality of data, and ensuring consistency and integrity of business-critical data and facts across the enterprise. This approach is known as the Information as a Service (IaaS) approach.

- **Information Integration**: This category of capabilities addresses the support of information integration and enables capabilities for information services.
- **Basic Information Management**: This category of capabilities addresses basic information management concerns such as metadata and unstructured data management.
- **Information Security and Protection**: This category of capabilities addresses the support of information security and protection concerns.
- **Business Analytics**: This category of capabilities addresses the support of business analytics and business activity monitoring. It enables organizations to leverage information to better understand and optimize business performance. It supports entry points of reporting to deep analytics and visualization, planning, aligned strategic metrics, role-based visibility, search-based access and dynamic drill-through, and alert and detect in-time actions.
- **Information Definition and Modeling**: This category of capabilities defines fundamental constructs of SOA information and events.
- **Information Repository**: This category of capabilities addresses support of the information repository in order to persist data such as metadata, master data, analytical data, operational data, and unstructured data.

This layer features the following capabilities:

Information Services

1. Ability to expose data as services, to add/remove/manipulate data entries in different services or service components, and to disable some data from outside access
2. Ability to interface with the Integration Layer in multiple ways such as message-based, service call, batch interface
3. Ability to handle representing data from various data sources in a unified data format; ability to transform and map data from one format to another and align data from different resources
4. Ability to manage the lifecycle of business entities
5. Ability to manage the hierarchy and relationship among data
6. Ability to validate records against defined business rules
7. Ability to validate and enforce data quality rules

8. Ability to notify and trigger actions based upon events detected within the data

Information Integration

9. Ability to perform Extract-Transform-Load (ETL capabilities) data from one source to other; ability to extract relevant information from sources, transform the information into the appropriate integrated form, and load the information into the target repository
10. Ability to perform Enterprise Information Integration (EII) capabilities, such as access to federated query to structure and unstructured data
11. Ability to virtualize data representing actual data from the actual data repositories of various types, such as a DB2 database in the Operational Systems Layer, or an Excel file
12. Ability to handle data transformation (including transformation of data types and contents) and to aggregate data from multiple data sources
13. Ability to perform data standardization and perform data reconciliation including semantic reconciliation
14. Ability to cleanse and match inbound records to existing data
15. Ability to cache data in support of the data virtualization/information services capability

Basic Information Management

16. Ability to manage and maintain metadata in a common metadata repository for the enterprise
17. Ability to capture, aggregate, and manage unstructured content in a variety of formats such as images, text documents, web pages, spreadsheets, presentations, graphics, email, video, and other multimedia
18. Ability to author, configure, manage, customize, and extend metadata

Information Security and Protection

19. Ability to handle access privileges of various participants to data
20. Ability to control access on individual data items
21. Ability to monitor and manage data usages using a log-like facility; typical traceability log includes: who has accessed the data, when, and what part of the data has been accessed

Business Analytics

22. Ability to analyze data access history and provide optimization algorithms and business intelligence for data optimization
23. Ability to query and search capabilities for enterprise information
24. Ability to visualize interactively the results from business analytics and data analysis

25. Ability to interface with the Integration Layer and obtain events from the Integration Layer; ability to analyze this event information, both in real-time/near real-time, as well as stored (warehoused) events

26. Ability to review and assess inbound service activity in the form of event information and determine responses or issue alerts/notifications

Information Definition and Modeling

27. Ability to define business vocabulary – glossary, terms, business entities

28. Ability to define a common information model as leveraged by IT such as entity relationships, logical data model for information repositories, and message model for service definition and specification

29. Ability to define business events

Information Repository

30. Ability to store operational and reshaped information (structured and unstructured) that adds business value including common model of data; used to share canonical forms (common data models) between SOA Integration Layer elements and also other SOA layer elements – typically invoked by other components (including information virtualization capabilities)

31. Ability to store instance and definition of master data and historical data that records changes to master data

32. Ability to store analytical data

15.1.3 Architecture Building Blocks (ABBs)

The ABBs responsible for providing these sets of capabilities in the Information Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1	Information Service	Information Services Gateway	1, 2
2		Data Aggregator	3
3		Data Validator	6
4		Information Lifecycle Manager	4
5		Hierarchy and Relationship Manager	5
6		Data Quality Manager	7
7		Quality of Service Layer: Event Manager	8
8	Information Integration	Information Integrity Manager	
9		Data Cleanser	14

#	Capability Category	ABB Name	Supported Capabilities
10		Data Rationalization Manager	13
11		Data Matcher	14
12		Data Virtualization Manager	
13		Data Representation Manager	11
14		Data Sourcing Manager	11
15		Data Cache	15
16		Integration Layer: Data Transformer	12
17		Data Consolidator	9
18		Data Federator	10
19	Basic Information Management	Information Metadata Manager	16
20		Content Manager	17
21		Master Data Authoring Environment	18
22	Information Security and Protection	Quality of Service Layer: Access Controller	19
23		Quality of Service Layer: Data-Driven Access Controller	20
24		Traceability Enabler/Auditor	21
25	Business Analytics	Data Miner	22
26		Query, Search, Reporting Engine	23
27		Analytics Visualization Engine	24
28		Quality of Service Layer: Business Activity Monitor	25
29		Quality of Service Layer: Business Activity Manager	25
30		Quality of Service Layer: Activity Correlation Manager	26
31	Information Definition and Modeling	Business Information: Business Glossary and Terms, Business Entities	27
32		Common Information: Entities and Data, Messages	28
33		Business Events	29
34	Information Repository	Data Repository	30-32

15.2 Details of ABBs and Supported Capabilities

15.2.1 Details of ABBs

This section describes each of the ABBs in the Information Layer in terms of their responsibilities.

15.2.1.1 Information Service Gateway

This ABB can be thought as a service container enforcing and supporting exposure of services, with all the associated supporting capabilities. In particular, it has three main responsibilities:

- To expose Information as a Service (IaaS)
- To manipulate data entries in different services and service components
- To control access to certain selected aspects of data; disable some data parts from outside access

This ABB acts as the gateway to the Information Layer. This ABB enables the hosting and exposure of information services by the SOA RA, forming a virtual data layer. It thus supports interfacing between the Information Layer and consumers of information services and is critical to expose Information as a Service (IaaS). It provides a consistent entry point to the Information Layer through multiple mechanisms such as messaging, service calls, and batch processing. This ABB leverages capabilities and ABBs from the Integration Layer.

15.2.1.2 Data Aggregator

This ABB is responsible for efficiently joining information – for example, structured and unstructured data – from multiple sources without creating data redundancy to help form a unified data view/model supported by the Data Virtualization Manager ABB.

Its responsibilities include:

- Dispatching requests to other ABBs in the Information Layer
- Invoking the Data Virtualization Manager ABB for handling data transformation (including transformation of data types and contents); and aggregating data from multiple data sources to provide a unified format and model to other ABBs and consumers of information services
- Invoking the Data Validator ABB to validate against business rules
- Invoking the Data Quality Manager ABB for enforcing data quality rules
- Invoking the Event Manager ABB in the Quality of Service Layer for triggering event notification based on data

15.2.1.3 Data Validator

This ABB is responsible for validating records against defined business rules.

15.2.1.4 Information Lifecycle Manager

This ABB is responsible to providing lifecycle management support for data; e.g., CRUD and to apply business logic based upon the context of that data.

15.2.1.5 Hierarchy and Relationship Manager

This ABB is responsible for managing the data hierarchies, groupings, relationships such as parent-child relationships, and relationships between enterprise data. This ABB is leveraged by the Data Virtualization Manager to build the relationships.

15.2.1.6 Data Quality Manager

This ABB is responsible for validating and enforcing data quality rules, standardizing the data for both value and structure, and performing data reconciliation including semantic reconciliation. It leverages the Information Integrity ABB to fulfill its responsibilities.

15.2.1.7 Quality of Service Layer: Event Manager

See Event Manager ABB in the Quality of Service Layer.

15.2.1.8 Information Integrity Manager

This ABB is responsible for data profiling, analysis, cleansing, data standardization, and matching. Data profiling and analysis services are critical for understanding the quality of data across enterprise systems, and for defining data validation, data cleansing, matching, and standardization logic required to improve data quality and consistency.

15.2.1.9 Data Cleanser

This ABB is responsible for cleansing and applying data quality rules. It enables detection and correction of corrupted or incorrect data.

15.2.1.10 Data Rationalization Manager

This ABB is responsible for performing data rationalizing and reconciliation.

15.2.1.11 Data Matcher

This ABB is responsible for matching inbound records to existing data. It supports deterministic matching and probabilistic matching of records.

15.2.1.12 Data Virtualization Manager

This ABB is responsible for providing virtual access and unified representation of enterprise data sources.

15.2.1.13 Data Representation Manager

This ABB is responsible for handling representation of data from various data sources in a unified data format and for creation of unified views of data. In other words, this ABB intends to hide various data sources and present data in uniform formats to other ABBs for data handling.

This ABB may link to various data sources and handle relationships between the data sources. This "virtualization" of the data makes consumers of information services (exposed through the Information Services Gateway) and other ABBs independent of the source and supports consistency in data.

15.2.1.14 Data Sourcing Manager

This ABB is responsible for enabling access to different data sources using different protocols. It provides unified access to data in files, databases, etc. It uses an Adapter ABB from the Integration Layer to provide the ability to integrate with data sources in different solution platforms (external data sources).

Examples may be relational sources (e.g., DB2, Oracle, or SQL Server databases), other structured data (e.g., Excel .CSV, web service request responses in XML format, and hierarchical stores on mainframes such as IMS), as well as unstructured data stores (such as images and documents). It manages interactions with the data sources in the Solution Platform and other SOA RA layers, but it is not responsible for addressing data and protocol transformation. This ABB represents the actual data repositories in various types, such as a DB2 database in the Operational Systems Layer, or an Excel file. It should be noted that this ABB in the Information Layer refers to high-level links associated with metadata to real data sources in the Operational Systems Layer. This ABB enables optimization of the data access by lazy loading or on-demand access of information. For example, instead of containing (e.g., attaching) a huge document, this ABB typically contains an on-demand link to the original document, together with some metadata describing the document (e.g., goals, purposes, and short descriptions) that help users decide whether they need to access the original document (e.g., a CEO may decide not to download a detailed design document while a project architect may decide to download and review). In addition, it should be noted this ABB typically represents industry-specific data structure; therefore, transformation may be needed for further processing.

15.2.1.15 Data Cache

This ABB is responsible for the caching of data in support of the data virtualization/information services capability. It enables addressing variations in temporal availability of data as well as improvement of performance. The variance in temporal availability of data is an issue associated with different data sources having different schedules for data being available; for example, one data source could be a time-based file feed, the other a mainframe batch program, and the third a real-time relational database. In such a scenario, for the consistent update and availability of data, it is useful to be able to cache it in some form. The data cache may use persistent data or non-persistent caching of data, which are implementation aspects.

15.2.1.16 Integration Layer: Data Transformer

See Data Transformer ABB in the Integration Layer.

15.2.1.17 Data Consolidator

This ABB is responsible for extracting relevant information from sources, transforming the information into the appropriate integrated form, and loading the information into the target repository. This ABB supports Extract-Transform-Load (ETL) from one or more source systems into a target system. It is also responsible for supporting real-time ETL capabilities with the

initial or incremental ETL of volume data into a target repository (e.g., data warehouse or master data repository).

15.2.1.18 Data Federator

This ABB is responsible for providing Enterprise Information Integration (EII) capabilities for federated query access to structured and unstructured data.

15.2.1.19 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

15.2.1.20 Quality of Service Layer: Data-Driven Access Controller

See Data-Driven Access Controller ABB in the Quality of Service Layer.

15.2.1.21 Traceability Enabler/Auditor

This ABB is responsible for monitoring and managing data usage using a log-like facility. It interprets log information and stores it in databases to analyze the data and initiate threat alerts. This ABB supports the ability to know who has accessed data, when it has been accessed, and what has been accessed and also supports data privacy through the obfuscation of sensitive data.

15.2.1.22 Information Metadata Manager

This ABB is responsible for managing and maintaining metadata in a common metadata repository for the enterprise, including structured and unstructured data; for example, metadata that describes the master data taxonomies and XML schemas and rules for business logic and data validation. It stores information regarding transformation of data types and content and the ability to aggregate data from multiple sources. It is used to share canonical forms (common data models) between SOA Integration Layer elements and other layers of the SOA RA. It supports, in particular, the ability to store, retrieve, and translate metadata into forms that can be effectively consumed by repositories local to other layers in the SOA RA. It facilitates re-use for metadata assets, semantics, models, templates, rules, etc. across the enterprise. Information integration capabilities are used to support the replication of changes to metadata that is contained in systems across the enterprise.

15.2.1.23 Content Manager

This ABB is responsible for capturing, aggregating, and managing unstructured content in a variety of formats such as images, text documents, web pages, spreadsheets, presentations, graphics, email, video, and other multimedia. It provides the ability to search, catalog, secure, manage, and store unstructured content to support the creation, revision, approval, and publication of content. It provides the ability to identify new categories of content and create taxonomies for classifying enterprise content. This ABB is also responsible for managing the retention, access control and security, auditing and reporting, and ultimate disposition of business records. It provides for the policy-driven movement of content throughout the storage lifecycle and the ability to map content to the storage media type based on the overall value of the content and context of the business content.

15.2.1.24 Master Data Authoring Environment

This ABB is responsible for authoring, configuring, approving, managing, customizing and extending master data as well as the ability to add or modify instance master data, such as product, vendor, and supplier. These services support the MDM collaborative style of use and may be invoked as part of a collaborative workflow to complete the creation, updating, and approval of the information for definition or instance master data.

15.2.1.25 Data Miner

This ABB is responsible for analyzing data access history as well as providing optimization algorithms and business intelligence for data optimization. It enables building of descriptive and predictive models by uncovering previously unknown trends and patterns in vast amounts of data from across the enterprise, in order to support decision-making.

15.2.1.26 Query, Search, Reporting Engine

This ABB is responsible for supporting *ad hoc* queries, search, reporting, slicing/dicing/drill-downs, and Online Analytical Processing (OLAP) capabilities for enterprise information.

15.2.1.27 Analytics Visualization Engine

This ABB is responsible for providing interactive visualization of analytics results and data analysis leading to better analyses, faster decisions, and more effective presentation of analytic results. It provides charting and graphing functionality, spatial dashboard reporting such as for scorecard reporting, spatial analysis, and rendering for interaction with components that provide user presentation.

15.2.1.28 Quality of Service Layer: Business Activity Monitor

See Business Activity Monitor ABB in the Quality of Service Layer.

15.2.1.29 Quality of Service Layer: Business Activity Manager

See Business Activity Manager ABB in the Quality of Service Layer.

15.2.1.30 Quality of Service Layer: Activity Correlation Manager

See Activity Correlation Manager ABB in the Quality of Service Layer.

15.2.1.31 Business Information

This ABB represents business vocabularies – business glossary and terms and key business entities of the organization and their definition.

15.2.1.32 Common Information

This ABB represents definition of entities and their relationship, logical data definition for database design, and message model for service definition and specification. Information is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm.

15.2.1.33 Business Event

This ABB represents definition of business events. Event is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm.

15.2.1.34 Data Repository

This ABB provides the essential foundation for the storage of operational and reshaped information that adds business value. The core data repositories are Analytical Data, Operational Data. Master Data, Unstructure Data, Metadata.

- **Analytical Data Repository**: Includes operational data stores, data warehouse, data mart, staging areas, and *ad hoc* workspaces.
- **Operational Data Repository**: Includes transactional hub, ERP, Supply Chain, CRM, etc.
- **Master Data Repository**: Stores instance and definition of master data and historical data that records changes to master data.
- **Unstructured Data Repository**: Includes textual objects, graphical objects, multi-media objects, etc.
- **Quality of Service Layer: Metadata Repository**: Enables storage of metadata. It stores metadata that describes the master data taxonomies and XML schemas and rules for business logic and data validation. It stores information regarding transformation of data types and content and the ability to aggregate data from multiple sources.

15.2.2 Structural Overview of the Layer

The ABBs in the Information Layer can be thought of as being logically partitioned into the following categories which support:

- Ability to support information services capability, critical to support a shared, common, and consistent expression of data
- Ability to integrate information across the enterprise in order to enable information services capability
- Ability to define metadata and master data that is used across the SOA RA and in particular the metadata that is shared across the layers
- Ability to secure and protect information
- Ability to support business analytics and business activity monitoring critical to the usage of the SOA RA and its realization
- Ability to define information and events that are some of the fundamental constructs of SOA
- Ability to persist and store data

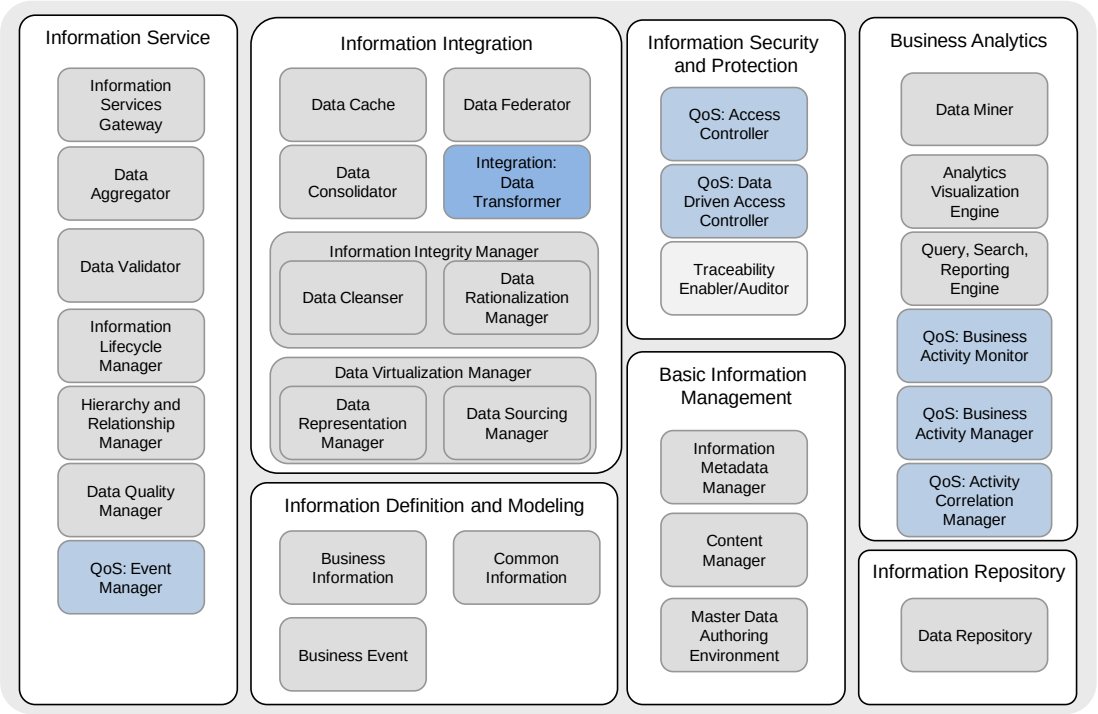


Figure 52: ABBs in the Information Layer

15.3 Inter-Relationships between the ABBs

The relationship among these ABBs is shown for different scenarios.

The first scenario is for Information as a Service (IaaS), where information is retrieved from multiple sources.

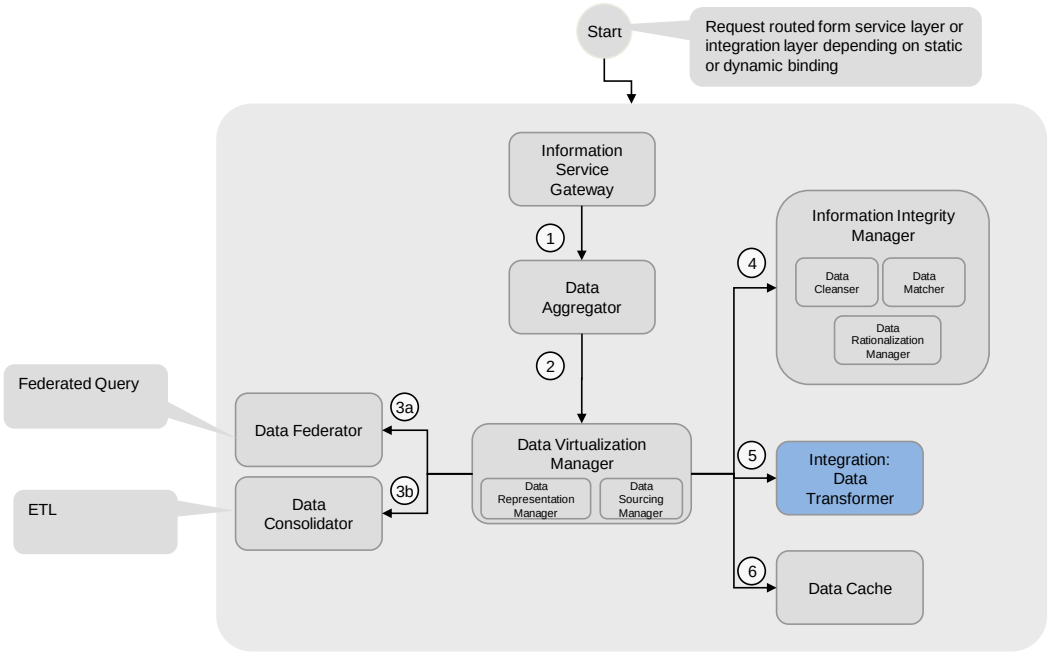


Figure 53: Key Interactions among ABBs in the Integration Layer in an IaaS Query Scenario

The second scenario relates to adding and updating information in the context of master data management.

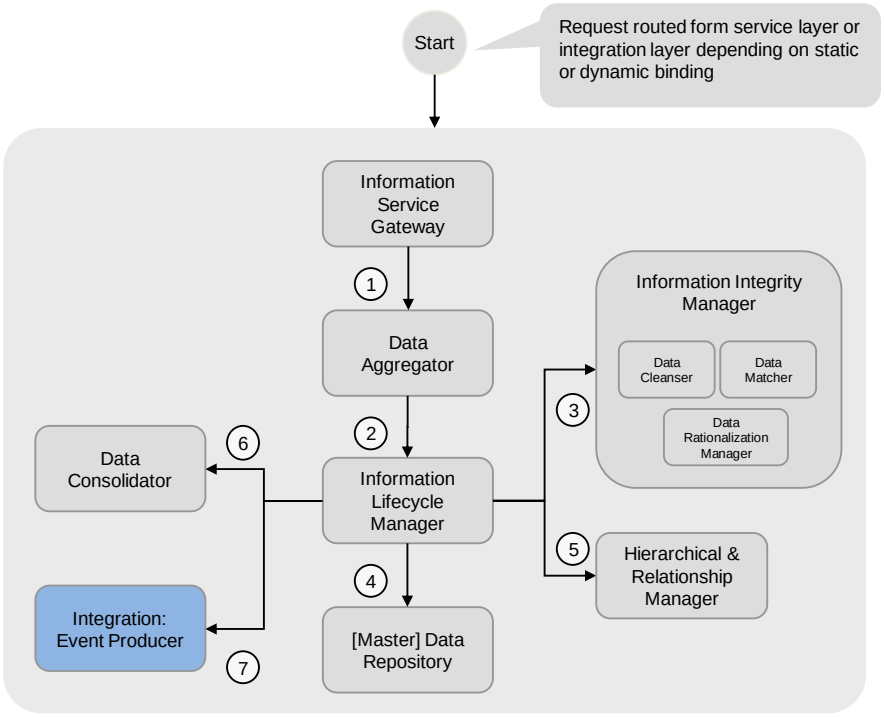


Figure 54: Key Interactions among ABBs in the Integration Layer for an Add/Update in an MDM Scenario

The third scenario is updating MDM by extracting deltas from source systems.

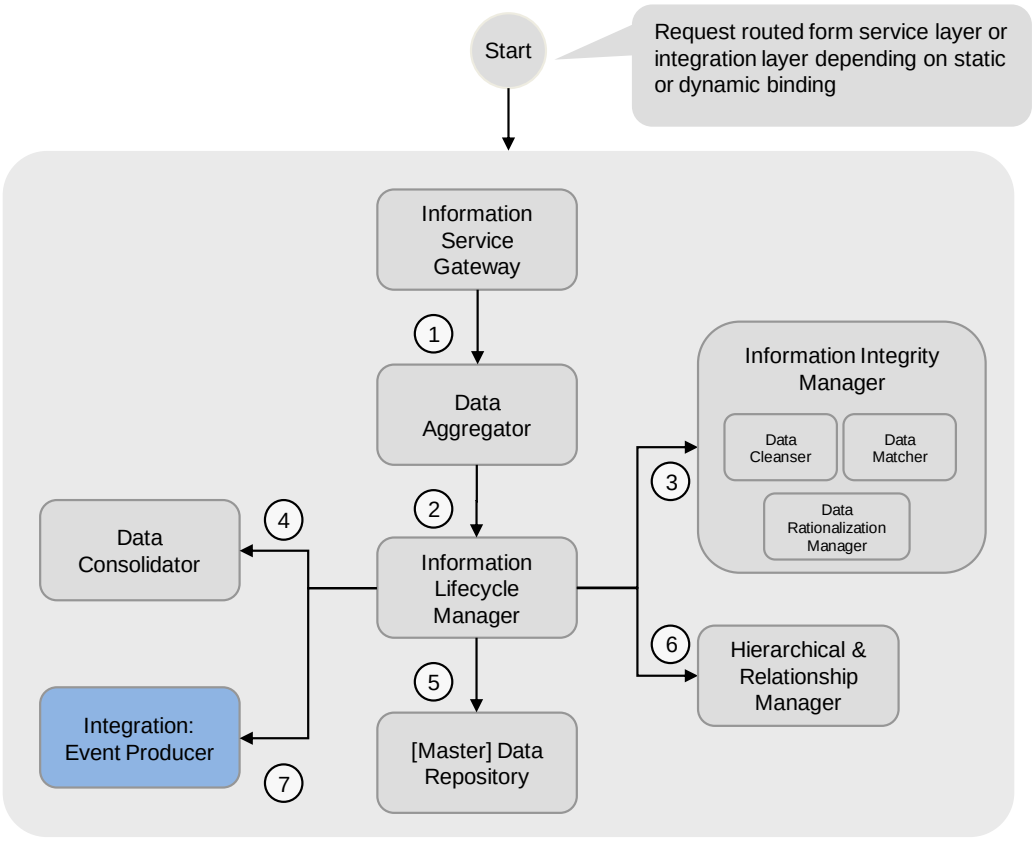


Figure 55: Key Interactions among ABBs in the Integration Layer for a Delta Extract and Update in an MDM Scenario

15.4 Significant Intersection Points with other Layers

Certain relationships exist between the ABBs in the Information Layer with those in other cross-cutting and horizontal layers:

- The Information Service Gateway ABB interacts with the Consumer Layer, Business Process Layer, Services Layer, Service Component Layer, Operational Systems Layer, Integration Layer, Quality of Service Layer, and Governance Layer.
- The Traceability Enabler/Auditor ABB interacts with the Quality of Service Layer.
- The Data Sourcing Manager ABB interacts with the Operational Systems Layer and the Governance Layer.

15.4.1 Interaction with Cross-Cutting Layers

- This layer leverages the Event Manager ABB in the Quality of Service Layer for notifying and triggering actions based upon events detected within the data. Events can be defined

to support data governance policies, based upon business rules, or can be time and date scheduled.

- This layer leverages the Data Transformer ABB in the Integration Layer for transforming and mapping of data from one format to another and aligning data from different resources.
- This layer leverages the Access Controller ABB in the Quality of Service Layer to enforce security policies and access privileges.
- This layer leverages the Data-Driven Access Controller ABB in the Quality of Service Layer to enforce access privileges on individual data items.
- This layer leverages the Business Activity Monitor ABB, Business Activity Manager ABB, and Activity Correlation Manager ABB in the Quality of Service Layer to monitor events, business activities, Key Performance Indicators (KPIs), to interface with the Integration Layer for event notification and propagation, and to analyze the event information, both in real-time/near real-time, as well as stored (warehoused) events, and decide responses to triggered events.
- The Policy Enforcer ABB from the Quality of Service Layer is leveraged by the Governance Layer to enforce governance policies, by all other layers to enforce security policies, and by the Integration Layer to enforce policies during mediation, by the Services Layer to enforce service policies, and by the Business Process Layer to enforce of polices on business processes.

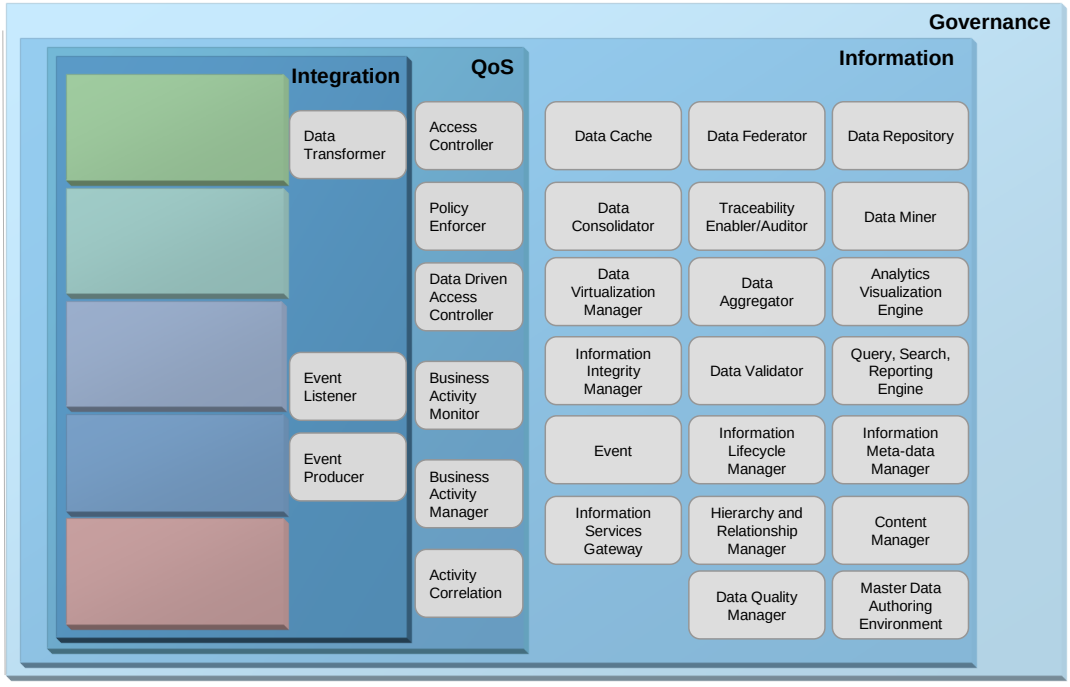


Figure 56: Key Interactions of the Information Layer with Cross-Cutting Layers

15.4.2 Interaction with Horizontal Layers

The four horizontal layers that are logically more functional in nature in the SOA RA – namely, Consumer Layer, Business Process Layer, Services Layer, and Service Component Layer – require information (structure and unstructured data, metadata, and messages) to fulfill their respective responsibilities and, therefore, rely on the Information Layer to access information. These horizontal layers are dependent on the ABBs of the Information Layer to fulfill their information needs.

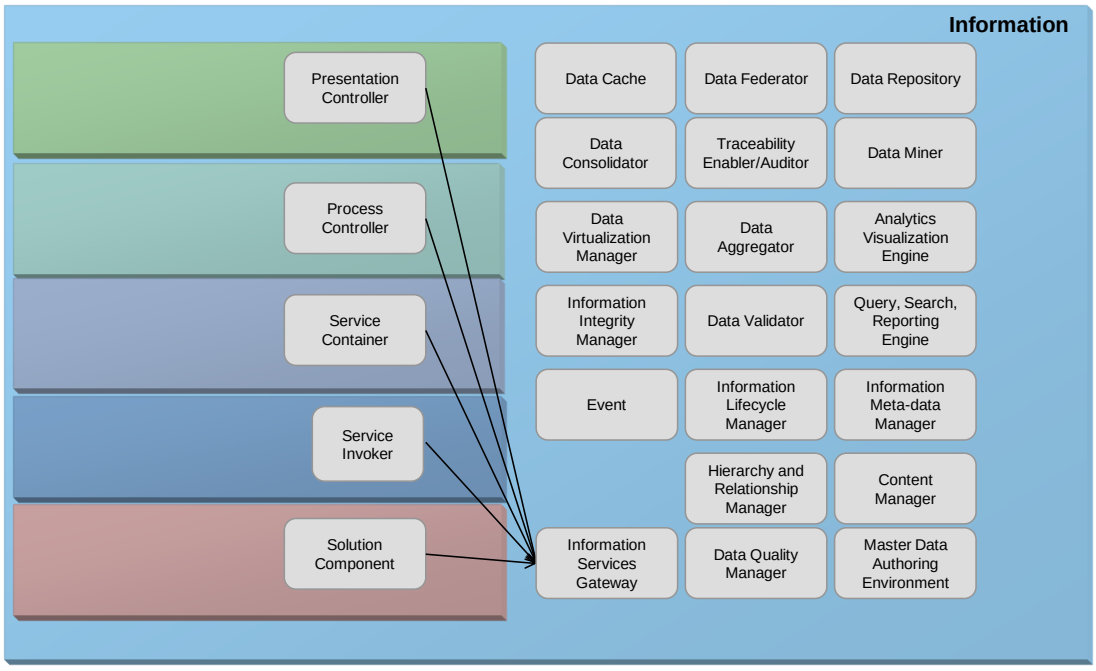


Figure 57: Key Interactions of the Information Layer with Horizontal Layers

15.5 Usage Implications and Guidance

Especially, for industry-specific SOA solutions, this layer captures all the common cross-industry and industry-specific data structures, XML-based metadata architectures (e.g., XML schema), and business protocols for exchanging business data. Some discovery, data mining, and analytic modeling of data are also covered in this layer. These common structures may be standardized for the industry or organization.

16 Governance Layer

16.1 Overview

SOA governance ensures that the services and SOA solutions within an organization are adhering to the defined policies, guidelines, and standards that are defined as a function of the objectives, strategies, and regulations applied in the organization and that the SOA solutions are providing the desired business value. SOA governance activities shall conform to corporate, IT, and enterprise architecture governance principles and standards. The Governance Layer will be adapted to match and support the target SOA maturity level of the organization.

The Governance Layer includes both SOA governance (governance of processes for policy definition, management, and enforcement) as well as service governance (service lifecycle). This covers the entire lifecycle of the services and SOA solutions (i.e., both design and runtime) as well as the portfolio management of both the services and SOA solutions managing all aspects of services and SOA solutions (e.g., Service-Level Agreement (SLA), capacity and performance, security and monitoring).

This layer can be applied to all the other layers in the SOA RA. From a Quality of Service (QoS) and management perspective, it is well connected with Quality of Service Layer. From a service lifecycle and design-time perspective, it is connected with the Services Layer. From an SOA solution lifecycle perspective, it is connected to the Business Process Layer.

The goal of the Governance Layer is to ensure consistency of the service and solution portfolio and lifecycles processes. In this layer, the extensible and flexible SOA governance framework will ensure that all aspects of SOA are managed and governed, such as:

- SLAs based on QoS and Key Performance Indicators (KPIs)
- Capacity and performance management policies
- Design-time aspects, such as business rules

The value of this layer is to ensure that the mechanisms are in place to organize, define, monitor, and implement governance from an enterprise architecture and solution architecture view.

16.1.1 Context and Typical Flow

This layer features the following characteristics:

- Defines policies, compliance, and exception characteristics
- Monitors the health of SOA services, solutions, and governance
- Reports on compliance, exceptions, service health, and versions

- Provides a consolidation point for business rules

The Governance Layer is aligned with the standardized SOA Governance Framework [25] which defines an SOA Governance Reference Model and SOA Governance Vitality Method. These provide definitions and best practices for defining a customized governance regimen for the enterprise.

The SOA Governance Reference Model includes definitions and best practices for governance guidelines, roles/responsibilities, governing processes, governed processes, and governance technology.

The governing processes are used to fulfill the responsibilities of the Governance Layer:

- Compliance processes are the conformance processes and policies including vitality which ensure the continued relevance of key governance model constructs, such as reference models, lifecycle (activities, work products), etc.
- Dispensation processes are processes and policies to handle exception to compliances.
- Communication processes disseminate and educate the fundamental constructs of lifecycle, governance models, etc.

The Governance Layer would have elements that facilitate and enable the implementation of the above governance processes.

Governed processes for the SOA solution are defined across all the other layers of the SOA RA, but can be articulated as:

- Service lifecycle management processes – describe the activities, roles, and work products to the modeling of services throughout the lifecycle, from identification to instantiation and retirement. These are often an extension of the organization's software development lifecycle.
- Solution lifecycle management processes – describe the activities, roles, and work products as they relate to the modeling of SOA solutions throughout the lifecycle, from identification to retirement.
- Service portfolio management processes – describe activities for selecting services to be developed and services to be re-used by solutions, ensuring an organization has the services appropriate to its needs. This influences the lifecycle management of those services.
- Solution portfolio management processes – describe activities for selecting solutions to be implemented and maintaining the correct mix of assets to support those solutions according the needs of the enterprise. Identifying services need by the solutions is one of the responsibilities and ties directly to input to the Service Portfolio Manager.

The SOA Governance Vitality Method phases, Plan/Define/Implement/Monitor, defined with best practices ensure that governance is a long-term process, keeping business and SOA solutions aligned:

- The Plan governance phase is responsible for analyzing, arranging, and scheduling solution-level monitoring and management.
- The Define governance phase is responsible for defining a solution-specific SOA-based governance control model and strategy.
- The Implement governance phase is responsible for enabling and realizing solution-level governance control.
- The Monitor governance phase is responsible for monitoring and managing solution-level system status according to predefined governance policies and plans.

These processes and tasks can be accomplished with a set of technical capabilities and Architecture Building Blocks (ABBs). ABBs implement the capabilities needed by the responsibilities and processes of the Governance Layer. Capabilities and ABBs will be needed for both dimensions of governance, the processes needed to create it, the governance vitality method, and the governance regimen itself. Capabilities and ABBs aligned with that standard are defined here.

The SOA Governance Vitality Method phases, Plan, Define, Implement, and Monitor, all require the capability to store and access governance information, define policies with a policy manager, and possibly develop and configure management tools. In addition, the Monitor phase needs the ability to monitor metrics, manage and enforce policies, and use change control and configuration management tools to react to policy changes. In addition, workflow can be used to implement the compliance processes.

The governance regimen – i.e., the customized compliance, dispensation, and communication processes to govern the SOA lifecycle and portfolio management – requires capabilities to store and access governance artifacts, manage and enforce policy, monitor metrics, and manage the configuration of the solution and governance. Change control may be needed to support changes to the system.

Governance applies to all phases of the SOA solution lifecycle, from architecture and design to implementation and maintenance.

Governance can be scoped to the enterprise as a whole, a line of business, a particular SOA solution, or a particular set of critical SOA processes and services.

16.1.2 Capabilities

There are a set of categories of capabilities that the Governance Layer needs to support in the SOA RA. These categories are:

- **Governance Planning**: This category of capabilities provides the ability to plan governance.
- **Governance Definition**: This category of capabilities provides the ability to define governance.
- **Governance Enablement and Implementation**: This category of capabilities provides the ability to implement governance.

- **SOA Metadata Storage and Management**: This category of capabilities enables storing and managing service metadata and governance artifacts.
- **Business Rule Definition and Management**: This category of capabilities provides the ability to define and manage business rules.
- **Policy Definition and Management**: This category of capabilities provides the ability to define and manage policies.
- **Monitoring**: This category of capabilities provides the ability to monitor application of policies, governance processes, and effectiveness of governance.
- **Management**: This category of capabilities provides the ability to manage governance artifacts and processes.
- **Workflow**: This category of capabilities provides the ability to capture and automate governance processes.

This layer features the following capabilities:

Governance Planning

1. Ability to analyze existing governance
2. Ability to identify governance goals, strategies, principles, and roles

Governance Definition

3. Ability to define governance processes
4. Ability to define governance artifacts

Governance Enablement and Implementation

5. Ability to enable governance
6. Ability to realize governance processes

SOA Metadata Storage and Management

7. Ability to store and search any kind of assets and artifacts
8. Ability to support the capture of service-related information at design time, and its dissemination to the other layers in the SOA in a standards-compliant interoperable manner
9. Ability to support the storage and dissemination of information supporting these capabilities:

   — Service contract definition (e.g., WSDL)

   — Service policy management

— Service version information management

— Service dependencies (e.g., the ability to integrate with a CMDB tool)

— Service management descriptions

— Canonical form and domain model specification for integration with the Information Layer

10. Ability to store governance artifacts
11. Ability to access governance artifacts
12. Ability to advertise/query for governance artifacts
13. Ability to advertise for services and metadata about services
14. Ability to find or query for services and metadata about services

Business Rule Definition and Management

15. Ability to capture, author, and define business rules
16. Ability to change, manage, and maintain business rules
17. Ability to store business rules

Policy Definition and Management

18. Ability to define policies
19. Ability to correlate business rules into policies
20. Ability to distribute policies
21. Ability to change, manage, monitor, and maintain current policies

Monitoring

22. Ability to monitor solution-level system status according to predefined governance policies and plans
23. Ability to manage solution-level system status according to predefined governance policies and plans
24. Ability to measure and gather metrics on the SOA services, SOA solutions, governing processes, and policy enforcement
25. Ability to evaluate metrics and test against policy regularly
26. Ability to use metrics to determine appropriateness of the current governance regimen
27. Ability to trigger checkpoints in the compliance process

28. Ability to indicate the status of governing and governed processes and their metrics in real-time

29. Ability to report the results of governing processes and their effectiveness

Management

30. Ability to do configuration management to implement and maintain governance

31. Ability to do change control to implement and maintain governance

32. Ability to access control and apply security policies for governance processes

Workflow

33. Ability to capture governing processes as workflow documents

34. Ability to automate governing processes

16.1.3 Architecture Building Blocks (ABBs)

These capabilities align with the SOA Governance Framework, SOA Governance Vitality cycle phases, and technology capabilities. These capabilities may need to be provided in an ongoing or cyclical basis as part of the SOA and SOA governance roadmap.

#	Capability Category	ABB Name	Supported Capabilities
1	SOA Metadata Storage and Management	Repository	7
2		Asset Repository	7
3		Service Repository	8-14
4		Service Registry	13, 14
5	Business Rule Definition and Management	Business Rule	15
6		Business Rules Manager	15-17
7		Business Rules Repository	17
8	Policy Definition and Management	Policy	18
9		Policy Manager	18-21
10		Quality of Service Layer: Policy Monitor	22
11	Monitoring	Quality of Service Layer: Monitor Metrics Tools	22-27
12		Dashboard	22, 23, 28, 29
13	Management	Reporting Tools	29
14		Development Tools	3-6

#	Capability Category	ABB Name	Supported Capabilities
15		Quality of Service Layer: Configuration Manager	30
16		Change Control Manager	31
17		Quality of Service Layer: Access Controller	32
18		Governance Gateway	30-32
19	Workflow	Workflow Process	33-34

Table 8: ABB to Capability Mapping for the Governance Layer

The same ABBs may be used by the SOA solution directly and to implement and execute the governance regimen.

16.2 Details of ABBs and Supported Capabilities

The purpose of this section is to identify solution ABBs that can be used to perform the SOA governing processes, compliance, dispensation, and communication. The same ABBs may be used to support both the governing and governed processes (e.g., a repository or a policy enforcement tool).

SOA governance ABBs are the technology to be used to enable governance and the whole or partial automation of the governing processes. The instantiation of these ABBs can range in ability from manual processes to sophisticated software.

16.2.1 Details of ABBs

16.2.1.1 Repository

This ABB represents a generic storage and enables organizations to organize, govern, and manage their valuable artifacts and assets scattered throughout the enterprise and to encourage re-use of existing assets within the organization. It provides the ability to search for the artifact and asset by different perspectives such as for general description, classification, usage, and content.

16.2.1.2 Asset Repository

This ABB enables organizations to organize, govern, and manage their valuable assets scattered throughout the enterprise and to encourage re-use of existing assets within the organization. It provides the ability to search for the asset by different perspectives such as for general description, classification, usage, and content. Assets could be business processes model, service artifacts, components design and code, models, documents, etc. Standards for asset repository and management include Re-usable Asset Specification (RAS) from OMG.

16.2.1.3 Service Repository

This ABB integrates with the Service Performance Manager ABB to support the runtime information collection and storage in order for users to evaluate service performance.

It acts as a design-time repository to store and locate metadata about services, including descriptions of the service contract, information about the QoS policies and security, versioning information, and runtime information such as end-points.

It is responsible for storing and accessing governance artifacts and best practices for SOA solutions and governance of SOA solutions from various perspectives, including organizational aspect, development aspect, runtime operational aspect, and lifecycle management aspect. Artifacts stored to guide governance may include service registrations, policy, guidelines, and governance processes.

This ABB integrates with the Service Registry ABB to support the runtime binding and service virtualization needs of SOA.

16.2.1.4 Service Registry

The ABB is responsible for allowing advertising and discovery of available services and supports the runtime binding of services and service virtualization. Think of this ABB as a runtime service repository. Services are available to the solution as well as governance processes. Advertisement of services should be governed. It contains metadata about services, including descriptions of the service contract, information about the QoS policies and security, versioning information, and runtime information such as end-points. This ABB may leverage the design-time Service Repository ABB to fetch metadata about services to fulfill the runtime needs of SOA such as dynamic/runtime binding and service virtualization. Standards for registries include UDDI.

This ABB contains service definitions at runtime and it plays a key role in service virtualization and service discovery. Service virtualization in this context is the exposure of a service end-point through a "proxy" (the registry). This in particular supports agility through service versioning so that the impact of services being released can be addressed through versions of services. The other aspect is the administration of the service where there are changes to the location of a service. Location is expressed in terms of an "end-point"; *aka*, the location from which a service is invoked. This could be the service container's address or some other unique identifier (URI).

16.2.1.5 Business Rule

This ABB defines a business rule constraining some aspects of business such as business processes, services, and information. Business rule is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm. The business rules may apply throughout the SOA solution and governance lifecycle.

16.2.1.6 Business Rules Manager

This ABB is responsible for defining, distributing, and maintaining business rules and their correlation to policies. The appropriate resulting policies are defined by using a Policy Manager ABB. This ABB supports rule implementation across multiple SOA RA layers.

16.2.1.7 Business Rules Repository

This ABB is responsible for storing and accessing business rules artifacts. This ABB is used by the Business Rule Manager ABB.

16.2.1.8 Policy

This ABB defines a policy describing principles to guide decisions to drive to desired outcomes. Policy is one of the fundamental constructs of an SOA solution and analysis and design based on the service-oriented paradigm. Policies may apply throughout the SOA solution and governance lifecycle. Policies could be at multiple levels, such as business level, architectural level, and/or operational level. It is important to capture the policy-related decisions and rules and policy information and its sources while defining policies such that the policies can be evaluated during policy enforcement by the Policy Enforcer in the Quality of Service Layer.

16.2.1.9 Policy Manager

This ABB is responsible for defining, authoring, distributing, and maintaining policies using policy management tools. Sophisticated policy managers may also check for and resolve policy conflicts. This ABB represents the primary Policy Administration Point in the SOA RA. This ABB is responsible for the distribution of the policies to one or more Policy Enforcement Points (PEPs) represented by the Policy Enforcer ABB in the Quality of Service Layer and its intersection with the other layers of the SOA RA for evaluation and enforcement purposes.

It is important to note that this ABB supports the management of policies needed to support security and logically includes all the responsibilities of a Security Policy Manager. A key tenant of an effective security program is management of security based on well-defined policies.

16.2.1.10 Quality of Service Layer: Policy Monitor

See Policy Monitor ABB in the Quality of Service Layer.

16.2.1.11 Quality of Service Layer: Monitoring Metric Tools

See Monitoring Metric Tools ABB in the Quality of Service Layer.

16.2.1.12 Dashboard

This ABB represents a set of tools that provide real-time status of the governing and governed processes and their metrics and checkpoints. It leverages the Monitoring Metric Tools ABB and Policy Enforcer ABB in the Quality of Service Layer.

16.2.1.13 Reporting Tools

This ABB represents a set of tools that customize, create, and generate reports on the execution of compliance and dispensation processes as well as the execution of checkpoints and monitoring of metrics. These reports can be stored with the Repository ABB. These reports can be created during any governance phase and any of the governance processes.

16.2.1.14 Development Tools

This ABB represents a set of tools used in the plan, define, implement phases of SOA governance to document governance policies and implement SOA governance metrics, checkpoints, and processes.

16.2.1.15 Quality of Service Layer: Configuration Manager

See Configuration Manager ABB in the Quality of Service Layer.

16.2.1.16 Change Control Manager

This ABB is used to control the updating of configuration, policies, and services. Change control applies to both the SOA solution and SOA governance. Change control processes are key processes to be governed.

16.2.1.17 Quality of Service Layer: Access Controller

See Access Controller ABB in the Quality of Service Layer.

16.2.1.18 Governance Gateway

This ABB is the gateway of all the ABBs in the Governance Layer to the other layers. In other words, it provides a focal point for processing both outgoing and incoming requests for governance management to and from the other eight layers.

16.2.1.19 Workflow Process

This ABB enables the automation of compliance and dispensation processes. The Workflow ABB from the Business Process Layer enables a business process to support manual intervention. This is often a requirement in a situation where error handling has to be performed. Standards include BPEL.

16.2.2 Structural Overview of the Layer

The Governance Layer applies to all the other layers of the SOA RA. ABBs of the Governance Layer can be thought of as being logically partitioned into categories that support:

- Ability to store and access (i.e., with registries, repositories, web sites, or databases) artifacts related to governance. Artifacts can be organization documentation, business process documentation, policies, compliance records, etc.
- Ability to define and manage business rules
- Ability to author and manage policies based on business rules at all phases of the SOA solution (design and runtime), including the ongoing governance vitality phase
- Ability to measure, monitor, and access governance metrics, both for ensuring governance policies are adhered to and for ensuring governance regimens continue to be appropriate
- Ability to manage and maintain governance, including the ability to do configuration, change control, security, and reporting

- Ability to automate processes and capture the processes in workflows

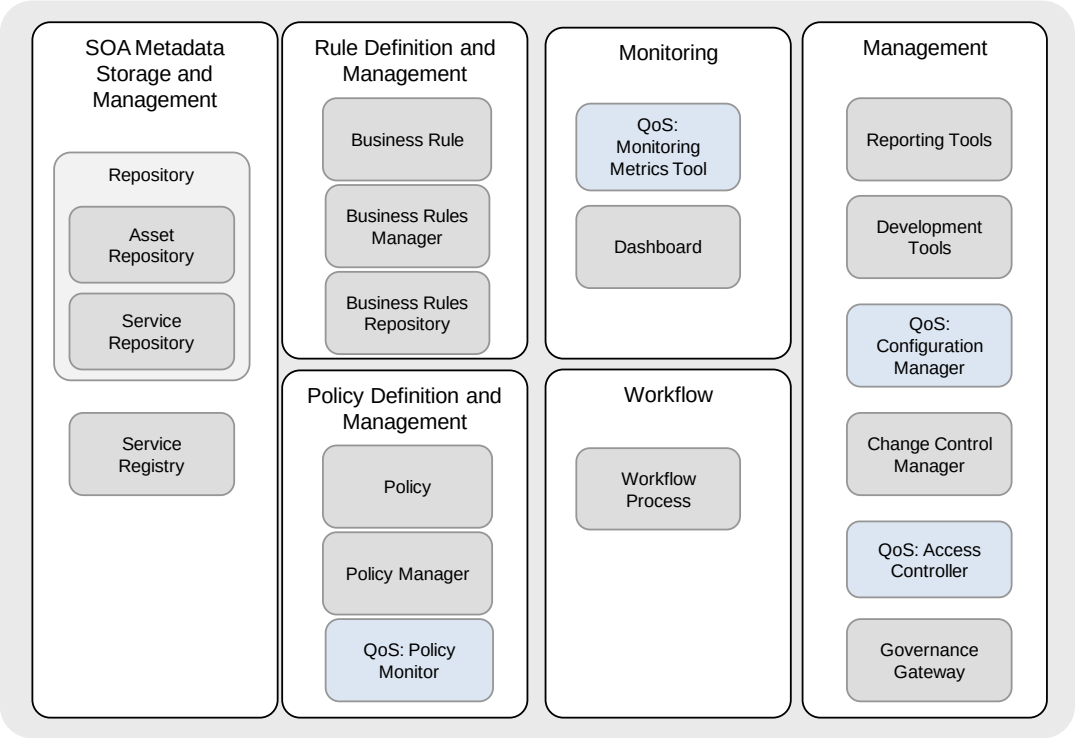


Figure 58: ABBs in the Governance Layer

The Governance Layer leverages ABBs from the Quality of Service Layer to fulfill its core responsibilities. The ABBs from the Quality of Service Layer leveraged by the Governance Layer are: Policy Monitor ABB, Monitoring Metrics Tool ABB, Configuration Manager ABB, and Access Controller ABB.

SOA governance capabilities and ABBs are used during the governing of an SOA solution. Other layers in the SOA RA define what technologies should be used to develop, deploy, and operate an SOA solution. The same technologies can be used to support the SOA solution and governance of the SOA solution.

SOA technology help realizing the SOA solution also needs governance, but the governance of these technologies is part of the service and solution portfolio and horizontal layers.

Each of the phases of governance will use a different set of ABBs.

A Plan Governance phase is responsible for analyzing, arranging, and scheduling solution-level monitoring and management; therefore, it would use the Service Repository ABB, Asset Repository ABB, and Business Rules Repository ABB to store governance information artifacts like governance principles, organization roles and responsibility, and business rules.

A Define Governance phase is responsible for defining a solution-specific SOA-based governance control model and strategy; therefore, it would use the Service Repository ABB and Asset Repository ABB to store the information artifacts, governance processes, and transition

processes for implementing governance. It may also use a Business Rules Manager ABB and Policy Manager ABB to author policies to be used in the implementation phase.

An Implement Governance phase is responsible for enabling and realizing solution-level governance control; therefore, it would use many of the governance ABBs, including the Service Registry ABB for governance information for services, Service Repository ABB for services and governance metadata, Policy Manager ABB to author policies, and the Quality of Service Layer: Configuration Manager to configure the Quality of Service Layer: Policy Enforcer ABB; Quality of Service Layer: Access Controller ABB to configure security policies and enforcement points.

A Monitor Governance phase is responsible for monitoring and managing solution-level system status according to predefined governance policies and plans; therefore, it would use the Quality of Service Layer: Policy Enforcer ABB, Quality of Service Layer: Monitoring Metrics Tool ABB, Dashboard ABB, and Reporting Tools ABB.

Each of the governing processes in the final governance regimen, compliance, dispensation, and communication, will use a different set of ABBs. An example compliance process is illustrated in the next section.

16.3 Inter-Relationships between the ABBs

Figure 59 illustrates the different ABBs and their inter-dependencies.

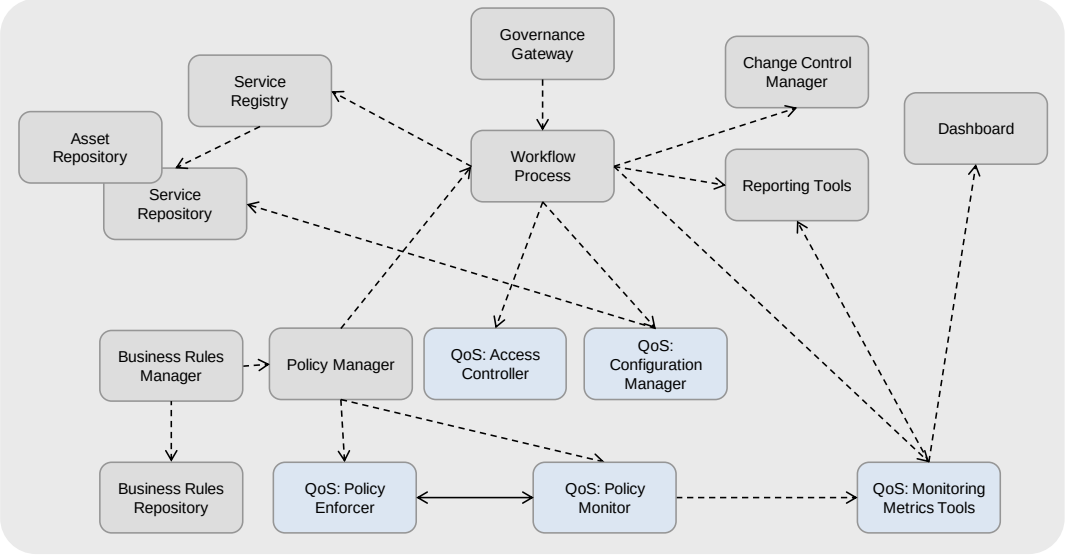


Figure 59: Relationships among ABBs in the Governance Layer

The Governance Gateway ABB invokes and governs the governing processes captured as workflows in the Workflow Process ABB.

The Workflow ABB captures and documents the governing processes. The workflows find governed services and other services to support governance using the Service Registry ABB. The Workflow ABB interacts with the Access Controller ABB, Policy Enforcer ABB, and Configuration Manager ABB in the Quality of Service Layer and Change Control Manager ABB

and Reporting Tools ABB in the Governance Layer to accomplish the goals of the governing process. The Workflow ABB also shares ongoing policy enforcement results with the Monitoring Metrics Tools ABB in the Quality of Service Layer which in turn shares policy enforcement results with the Dashboard ABB and Reporting Tools ABB.

The Business Rule Manager defines policies in the Policy Manager ABB. The Policy Manager ABB shares policies with the Asset Repository ABB or Service Repository ABB, Policy Enforcer ABB in the Quality of Service Layer, Access Controller ABB in the Quality of Service Layer, and governing process workflows.

The Policy Enforcer ABB in the Quality of Service Layer and governing process Workflow ABB shares policy results with the Monitoring Metrics Tool ABB in the Quality of Service Layer.

Figure 60 shows a flow for an example compliance process:

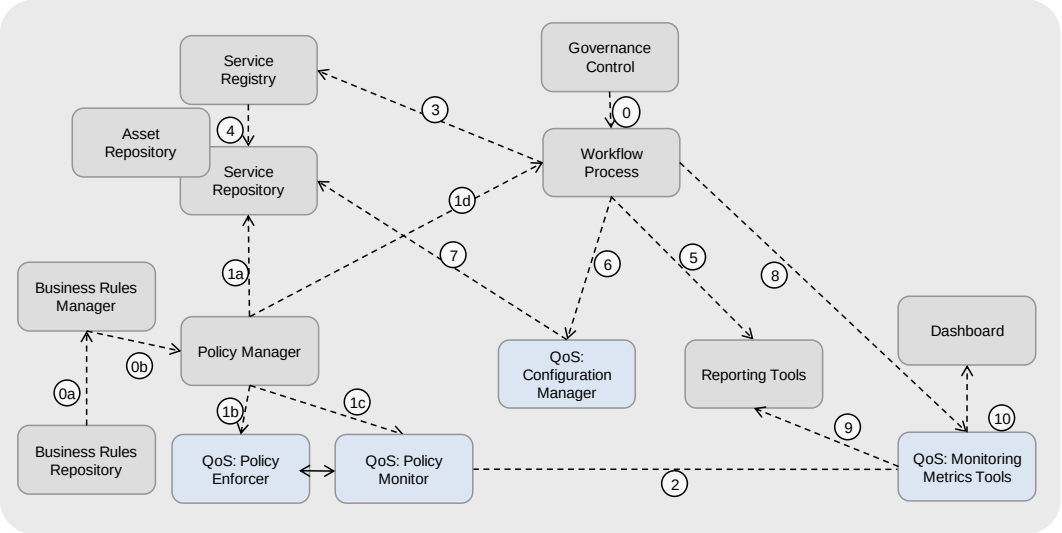


Figure 60: Sample Interactions among ABBs in the Governance Layer for a Governance Compliance Process

In this example, a business rule has indicated that a policy must be set that services that have excessive failures shall be inactivated and operations notified via the Dashboard ABB. At (1a-d) the policy to inactivate a service after five failures in a day is defined and distributed to the service repository, policy monitor, policy enforcer, and compliance process workflow. When the policy monitor detects that the service has failed more than five times, it invokes the policy enforcer to inactivate the service and notifies the monitoring metrics tool. The compliance process is running and at (3) looks up the service information which retrieves the policies for the service from the repository using (4). Now the compliance process checks with the monitoring metrics for policy exceptions and finds that the service has exceeded its failure threshold. The compliance process interacts with the configuration manager to configure the service to be out-of-use, the configuration manager interacts with the status manager is the Quality of Service Layer to update the repository with the current service configuration set to out-of-use. Now the workflow reports that the service has been taken out of use, as does the monitoring metrics. The

monitoring metrics update the dashboard with a new status for the service being out of use. Here it is clear that the Governance and Quality of Service Layers work cooperatively.

16.4 Significant Intersection Points with other Layers

The Governance Layer is related to all the other layers of the SOA RA. All of the horizontal and vertical layers have assets that are governed by the Governance Layer. Some of the layers, especially Quality of Service, Integration, Services, and Business Process Layers contain ABBs that the Governance Layer needs to leverage.

16.4.1 Interaction with Cross-Cutting Layers

The Governance Layer is responsible for governing all the other cross-cutting layers, namely, Integration, Information Architecture, and Quality of Service Layers. In addition, the Governance Layer relies on ABBs in the Quality of Service Layer to fulfill its core responsibilities.

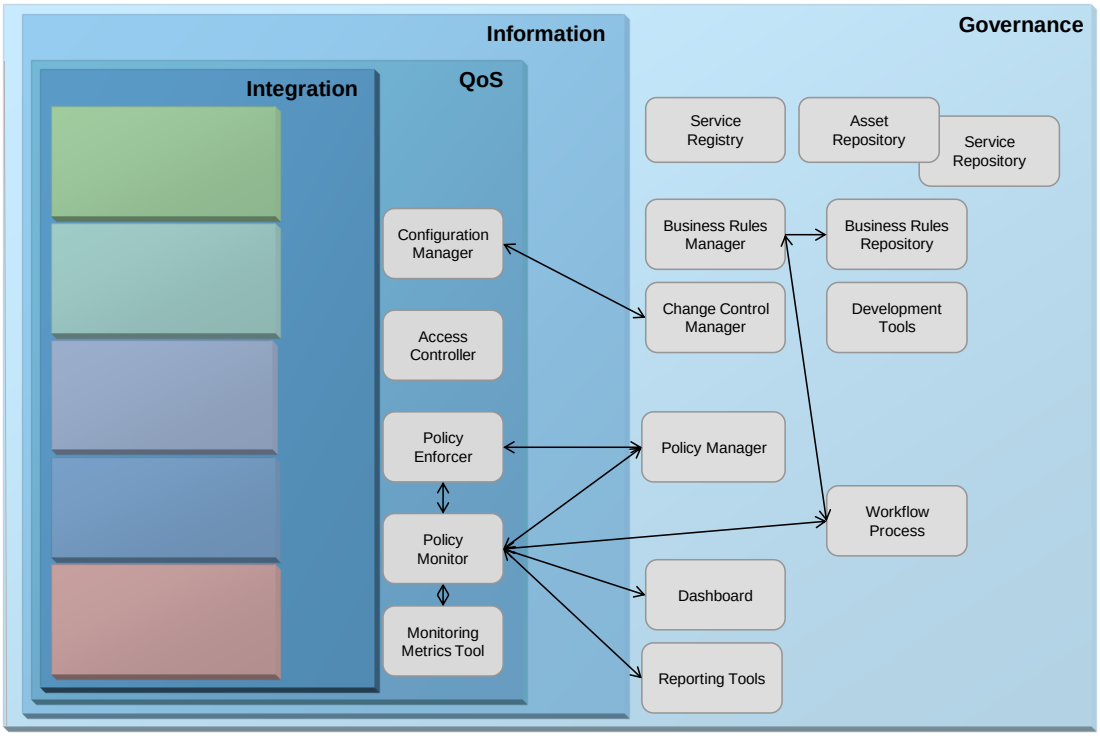


Figure 61: Key Interactions of the Governance Layer with Cross-Cutting Layers

Governance defines the policies used to drive the aspects of QoS in the Quality of Service Layer. The Governance Layer is dependent on the Quality of Service Layer to provide the following capabilities:

- The Business Rule Manager ABB is needed to ensure that the appropriate policies are set to support the policy manager capability. It also supports the Policy Enforcer ABB in the Quality of Service Layer in fulfilling its responsibilities through the Policy Manager ABB.

- It leverages the Policy Enforcer ABB in the Quality of Service Layer to enforce policies related to change control processes as required by the Change Control Manager ABB to ensure that change control is performed appropriately. Similarly, it leverages the Policy Enforcer ABB in the Quality of Service Layer to enforce security policies and policy to configure the solution using the Configuration Manager ABB in the Quality of Service Layer. It also leverages the Policy Enforcer ABB in the Quality of Service Layer to enforce governance policy and monitoring metrics for a solution.

- It leverages the Access Controller ABB in the Governance Layer to define the policies used to configure the security for the SOA solution and the governing processes via the security manager capability.

- It leverages the Configuration Manager ABB in the Quality of Service Layer in solution configuration and changing governance workflow processes. The Governance Layer defines the policies used to configure the solution using the Configuration Manager ABB in the Quality of Service Layer. In case there are automated governing process workflows, the workflow adjusts and changes the configuration using the Configuration Manager ABB in the Quality of Service Layer in order to adhere to the governance policies.

- It leverages the Monitoring Metrics Tools ABB and Policy Enforcer ABB in the Quality of Service Layer to measure, gather, evaluate, and test metrics against policies on a regular basis. The Governance Layer defines the policies used to implement the monitoring of metrics of the Governance Layer and SOA solution. Metric monitoring and analysis is used to drive governing processes and workflows to correct any policy violations and use the dashboard. Metrics and policy exceptions are also used to drive re-evaluation of the current governance regimen. Metrics are gathered on SOA services, governed processes, and governing processes. The Dashboard ABB leverages the Monitoring Metrics Tools ABB in Quality of Service Layer to customize what metrics and events should be visible on the governance and solution dashboard.

16.4.2 Interaction with Horizontal Layers

Governed assets exist in all of the horizontal layers; for example:

- IT infrastructure and implementation assets are governed in the Operational Systems Layer.

- Enterprise component are governed in the Service Component Layer.

- Services are governed in the Services Layer. Policies defined by governance decide which services will be built and re-used.

- Business processes are governed in the Business Process Layer.

- Governance and integration intersection – policies defined by governance – govern the mediation of interactions. The Integration Layer will utilize the registry and repository to actually find an end-point as a result of service invocation.

These horizontal layers define the different kinds of policies by interfacing with the Policy Manager ABB. In addition to governing these horizontal layers, the Governance Layer uses the

Business Process Layer to capture governance process and the Workflow ABB leverages the Business Process Layer to define the governance processes.

The Services Layer leverages the Service Repository ABB to store service definition/contracts, policy, and metadata about services during design time. The Services Layer leverages the Service Registry ABB to store service definition/contracts and policy and metadata about services to be used during runtime in order to discover services and bind to service provider/end-point enabling service virtualization.

The Integration Layer leverages the Service Registry ABB to determine the end-point for a service request and to enable service virtualization.

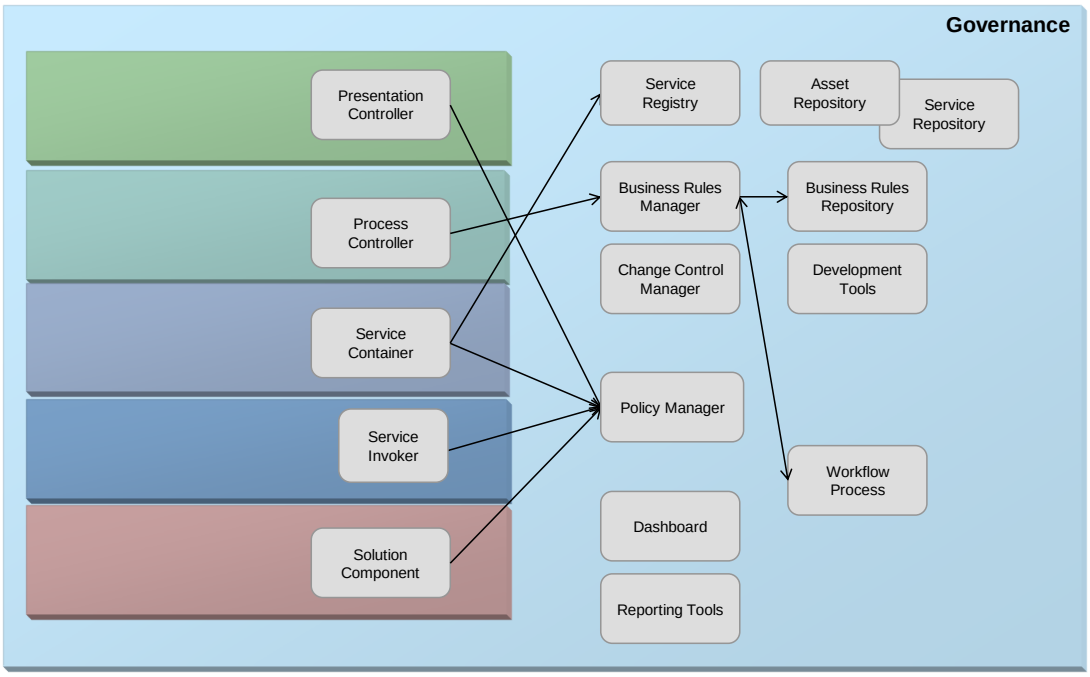


Figure 62: Key Interactions of the Governance Layer with Horizontal Layers

In summary, the Governance ABBs are used by other SOA RA layers:

- The Service Repository ABB is used by the Services, Quality of Service, and Integration Layers.
- The Service Registry ABB is used by the Services, Business Process, Consumer, Integration, and Quality of Service Layers.
- The Policy Manager ABB is used by the Integration and Quality of Service Layers.
- The Business Rule Manager ABB is used by the Business Process, Services, Service Component, and Quality of Service Layers.

16.5 Usage Implications and Guidance

At the heart of these processes is the Service Model, the unifying concept that binds these elements to together and makes them relevant.

16.5.1 Options and Design Decisions

Four of the design decision points that exist are:

- The use of a standard service registry and repository *versus* roll-your-own
- Collaboration technologies for communication and vitality
- Automation of service lifecycle and tracking
- Automation of compliance and exception handling processes

The information in this layer that is collected and made available via a repository consists of, for example:

- Guidelines for SOA governance
- Guidelines for service and SOA solution lifecycle and portfolio management
- Best practices
- Business rules
- Policies (e.g., security)
- Standards
- Service and SOA solution roadmaps
- Compliance, dispensation, and communication documentation

As a result, it is necessary that the Governance Layer capabilities and ABBs support all of governing all of these processes. The governance of each of these processes may require different ABBs.

Another important responsibility of the Governance Layer is to measure, gather, evaluate, and test metrics against policies on a regular basis. KPIs for this layer may include:

- Usage metrics of a service
- Downtime and failure statistics on a service or service set
- Policy violations
- Number of services compliant and number applied for exceptions

17 Related Work and Usages of the SOA RA

The SOA Reference Architecture (SOA RA) provides a mechanism for use in a variety of scenarios:

- For organizations adopting SOA
- For organizations building SOA components (SOA product vendors)
- For organizations providing services in the construction of SOA (integrators)
- For organizations building standards centered around the specification of SOA standards

For organizations adopting SOA, it provides a number of uses, including using the SOA RA to create SOA solutions including:

- Business process-driven
- Tool-based architecture-driven
- Message-based application integration through:
  - — Service-oriented integration
  - — Data access-based (information or data services)
  - — Legacy encapsulation and wrapping
- Legacy componentization and transformation

As we apply an SOA modeling and delivery method, every element of SOA that is identified is mapped back to the SOA RA providing a "dashboard view" of the SOA in progress; useful as a communication means for various business and IT stakeholders.

In addition, the SOA RA is used to define capabilities and the technical feasibility of solutions. This usage is focused on a realistic technique of identifying key technical extensible prototypes that test the premises of the architecture and its decisions in a risk-driven fashion.

The SOA RA provides a checklist of key elements that must be considered when you build your SOA: mandatory as well as optional layers, attributes, Architecture Building Blocks (ABBs), design decisions, and interaction patterns.

It is important to recognize that SOA solutions are designed and implemented by leveraging existing techniques and technologies. They have an associated set of best practices that are not specifically related to SOA. For example, writing robust J2EE applications and components is an important part of building SOA solutions. In this specification, we focus for the most part on the areas that are Critical Success Factors (CSFs) in building SOAs.

The SOA RA applies to various types of practitioners such as Enterprise Architects, Solution Architects, etc. The SOA RA is an abstract, logical design of an SOA. Thus, it answers the question: "What is an SOA?". Architects can use it as a checklist of layers, ABBs, and their relations in each layer, the options available, and decisions that need to be made at each layer. The layers provide a starting point for the separation of concerns needed to build an SOA.

A recurring theme in the context of SOA projects has been the applicability of SOA within multiple areas of increasing scope: a single project, a line of business, a few lines of business-sharing services, enterprise scale, supply-chain (value-net), and a larger SOA ecosystem. In each case the principles of SOA tend to be applied in a similar manner. This *self-similarity* of the application of SOA concepts recursively within larger or smaller scopes is termed "fractal" usage of the SOA paradigm.

When we apply SOA, as defined in the SOA RA, to a given level of granularity of an SOA ecosystem, we will typically find the need to create the same layers for each level of granularity. Thus, enterprise architecture might use the SOA RA as an SOA solution template that will be customized or instantiated for each line of business or each product line (depending on how the organization is structured). To participate in an SOA or services ecosystem, a company would need to have a standard reference architecture such as that depicted by the SOA RA, in order to facilitate the integration and collaboration of architectures across companies. Thus, standardization would benefit companies at the architectural level just as it has benefited them at the level of data interchange via XML and XML Schema.

A Relationship to Other SOA Standards

The "Navigating the SOA Open Standards Landscape Around Architecture" joint White Paper from OASIS, OMG, and The Open Group [22] was written to aid the SOA community navigate the myriad of overlapping technical products produced by these organizations with specific emphasis on the "A" in SOA; i.e., Architecture.

This joint White Paper explains and positions standards for SOA reference models, ontologies, reference architectures, maturity models, modeling languages, and standards work on SOA governance. It outlines where the works are similar, highlights the strengths of each body of work, and touches on how the work can be used together in complementary ways. It is also meant as a guide to users of these specifications for selecting the technical products most appropriate for their needs, consistent with where they are today and where they plan to head on their SOA journeys.

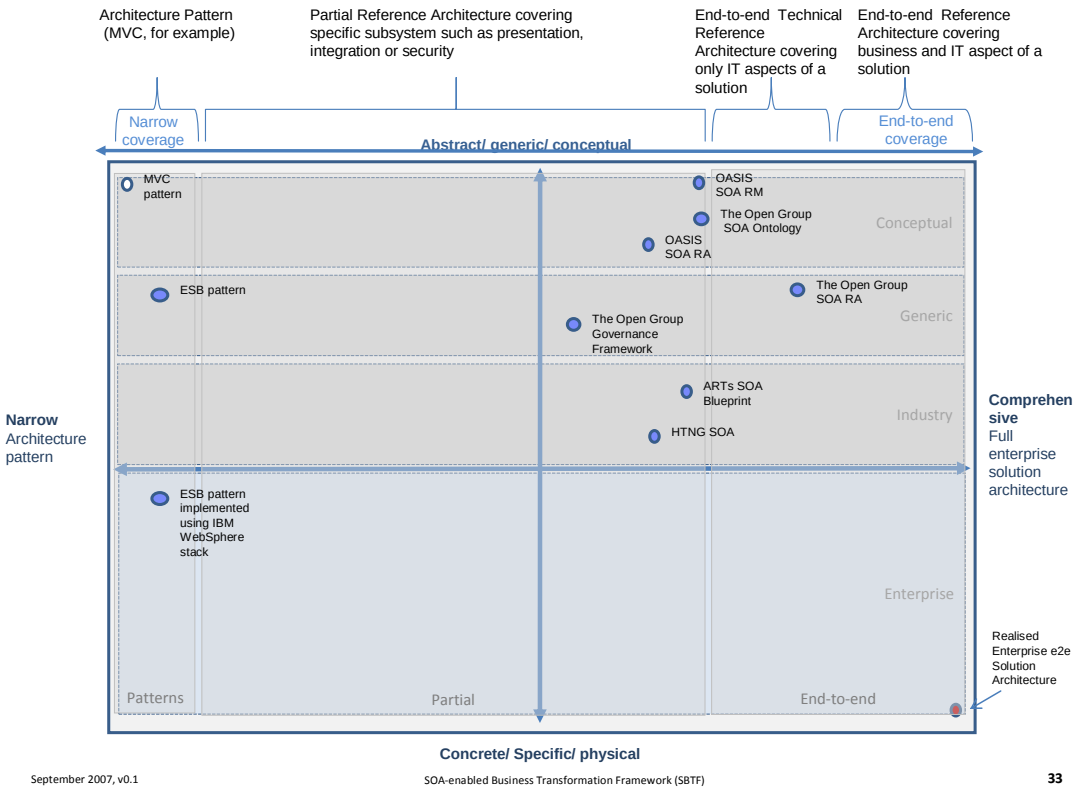
While the understanding of SOA and SOA governance concepts provided by these works is similar, the evolving standards are written from different perspectives. Each specification supports a similar range of opportunity, but has provided different levels of detail for the perspectives on which they focus. Therefore, although the definitions and expressions may differ somewhat, there is agreement on the fundamental concepts of SOA and SOA governance.

The following is a summary of the positioning and guidance on the specifications:

- The OASIS Reference Model for SOA (SOA RM) is the most abstract of the specifications positioned. It is used for understanding core SOA concepts. (See [23].)
- The Open Group SOA Ontology extends, refines, and formalizes some of the core concepts of the SOA RM. It is used for understanding core SOA concepts and facilitates a model-driven approach to SOA development. (See [24].)
- The OASIS Reference Architecture for SOA Foundation is an abstract, foundation reference architecture addressing the ecosystem viewpoint for building and interacting within the SOA paradigm. It is used for understanding different elements of SOA, the completeness of SOA architectures and implementations, and considerations for cross-ownership boundaries where there is no single authoritative entity for SOA and SOA governance. (Refer to: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf.)
- The Open Group SOA Reference Architecture (this document) is a layered architecture from a consumer and provider perspective with cross-cutting concerns describing those Architecture Building Blocks (ABBs) and principles that support the realizations of SOA. It is used for understanding the different elements of SOA, deployment of SOA in the enterprise, the basis for an industry or organizational reference architecture, implication of architectural decisions, and positioning of vendor products in SOA context.
- The Open Group SOA Governance Framework is a governance domain reference model and method. It is for understanding SOA governance in organizations. The OASIS

Reference Architecture for SOA Foundation contains an abstract discussion of governance principles as applied to SOA with particular application to governance across boundaries. (See [25].)

- The Open Group SOA Integration Maturity Model (OSIMM) is a means to assess an organization's maturity within a broad SOA spectrum and defines a roadmap for incremental adoption. It is used for understanding the level of SOA maturity in an organization. (See [21].)
- The Object Management Group SoaML Specification supports services modeling UML extensions. It can be seen as an instantiation of a subset of The Open Group SOA Reference Architecture used for representing SOA artifacts in UML. (Refer to: www.omg.org.)



Fortunately, there is a great deal of agreement on the foundational core concepts across the many independent specifications and standards for SOA. This could be best explained by broad and common experience of users of SOA and its maturity in the marketplace. It also provides assurance that investing in SOA-based business and IT transformation initiatives that incorporate and use these specifications and standards helps to mitigate risks that might compromise a successful SOA solution.

It is anticipated that future work on SOA standards may consider the positioning in this document to reduce inconsistencies, overlaps, and gaps between related standards and to ensure that they continue to evolve in as consistent and complete a manner as possible.

B Relationship to Open Group Guide: Using TOGAF to Define and Govern SOA

The Open Group Guide, Using TOGAF to Define and Govern Service-Oriented Architectures [26], describes one of the ways in which to instantiate the SOA Reference Architecture (SOA RA) using TOGAF. In that description, the practical guide suggests the use of the TOGAF meta-model to model the SOA RA adapted to an organization.

This mapping between the TOGAF meta-model and the meta-model used in the SOA RA is described below.

SOA RA Entity	TOGAF Entity
Capability	Platform Service
ABB	Logical Technology Component
Interaction	Contract
Enabling Technology	Physical Technology Component
Information Model	Information Component Model
NFR	Service Quality
KPI	Measure

Index