Snapshot

SOA Reference Architecture

Snapshot

SOA Reference Architecture

Published by The Open Group, March, 2011

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

c.harding@opengroup.org

Contents

List of Figures

List of Tables

Preface

The Open Group

The Open Group is a vendor-neutral and technology-neutral consortium, whose vision of Boundaryless Information Flow™ will enable access to integrated information within and between enterprises based on open standards and global interoperability. The Open Group works with customers, suppliers, consortia, and other standards bodies. Its role is to capture, understand, and address current and emerging requirements, establish policies, and share best practices; to facilitate interoperability, develop consensus, and evolve and integrate specifications and Open Source technologies; to offer a comprehensive set of services to enhance the operational efficiency of consortia; and to operate the industry's premier certification service, including UNIX® certification.

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

The Open Group has over 15 years' experience in developing and operating certification programs and has extensive experience developing and facilitating industry adoption of test suites used to validate conformance to an open standard or specification.

More information is available at www.opengroup.org/certification.

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

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards-compatible and those which are not.

A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.

A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

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 not the final Technical Standard for SOA Reference Architecture. Its purpose is to provide a Public Draft for Public Comment. It has been developed by the SOA Reference Architecture project of the SOA working group within The Open Group. It does not represent the consensus of The Open Group members.

Trademarks

Acknowledgements

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

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

In addition, The Open Group would like to record its appreciation of the inputs and advices from the following people from IBM: Raj Cherchatil, Arnauld Desprets, Jorge Diaz, Marc Fiammante, Donald Ferguson, Greg Flurry, Rolando Franco, Biffle French, George Galambos, Andrew Hately, Rob High Jr., Kerrie Holley, Joe Hardman, Petra Kopp, Rao Kormili, David Janson, Min Luo, Stefan Pappe, Emily Plachy, Siddarth Purohit, Robert Rafuse, Rachel Reinitz, Rick Robinson, Tony Storey, Raghu Varadan, Dan Wolfson, Bobby Wolf, Jamshid Vayghan, Olaf Zimmerman and other members of the IBM worldwide SOA community of practice.

From ApTSi we would like to record our appreciation of input and advice from Dinakar Sosale, Thomas Osberg, Anne Bonnar, Neb Brankovic and John Mathew

Referenced Documents

The following documents are referenced in this Document Type:

To be written

1 Introduction

1.1 Objective

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

The usage of the SOA Reference Architecture is a key enabler for the achievement of the value propositions of a Service-oriented Architecture.

This specification presents a SOA Reference Architecture (SOA RA), which provides guidelines and options for making architectural, design, and implementation decisions in the implementation of solutions. The goal of this SOA Reference Architecture standard 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 Reference Architecture is to answer some of the key questions and issues encountered by architects, including but not restricted to common questions such as these:

- "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 a 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

[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.

development life-cycle. These facilitate and ultimately enable automation and streamlining the process of modelling and documenting the architectural layers, the capabilities and the architectural 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 a SOA. It is 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 specification we present the results of abstracting and using a SOA Reference Architecture based on multiple projects in different industries across the world.

During these projects, a number of key underlying themes have come up that have been formalized into a meta-model for the SOA Reference Architecture (see figure xxx).

This meta-model defines a number of layers, architectural building blocks, 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 Reference Architecture provides a common systems architecture [9], and the mechanism to translate it to an industry architecture and an 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 an SOA's components are and how to pick specific solutions, and a mechanism to align technology with business requirements.

The SOA Reference Architecture 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 architectural building blocks (ABB). 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 typing 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 Reference Architecture then provides the ABBs that will realize those capabilities thus acting as a mechanism for initially deferring and then explicitly making implementation decisions as indicating how a given component of the solution, the ABB will choose to be implemented on their project.

The SOA Reference Architecture is based on establishing the building blocks of SOA: services, components and flows (processes) that collectively support business processes and goals. The meta-data 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 of reusability when designing and developing solution assets for rapid development, deployment and management of SOA solutions within industry or cross industries.

This document is organized as follows.

- Chapter 1 provides a general introduction.
- Chapter 2 provides an overview of the principles on which the SOA RA was derived
- Chapter 3 provides an introduction to the SOA Reference Architecture
- Chapter 4 over views the Reference Architecture
- Chapter 5 reviews the layers of the Reference Architecture in detail
- Chapter 6 illustrates applications of the Reference Architecture

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 Reference Architecture will identify the architectural building blocks and associated capabilities in each of the layers. It will also include examples of application of the reference architecture. 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 [8] (PMS notation and the birth of ADLs), Shaw and Garlan on Software Architecture [6], 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 [10]. This can be visualized as a set of components (we will call architectural building blocks to distinguish the generic nature of these), connectors, constraints [6], composition, containers and configurations. Arsanjani refers to these as the 6C's of an architectural style [7]. 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 [5] 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 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 services provided are published and consumers are insulated from the details of the underlying implementation by a provider.

Thus, a 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 goals. Not only does it provide flexible decoupled functionality that can be reused, but also provides the mechanisms to externalize variations [7] of quality of service in declarative specifications such as WS-Policy [11] and related standards.

2.1.1 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 solutions 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 business goals to IT, 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 reuse over time.

However, significant challenges in creating a SOA solution still remain. Correct design, solution element selection and combination, modelling of services, governance, interoperability, and the ability to identify different components are 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 a SOA solution?"
- "How can a SOA solution be organized as an architectural framework with inter-connected architectures and transformation capabilities?
- How can a SOA solution be designed in a manner that maximizes asset reuse?
- How can automated tools take the guess work out of architecture validation and capacity planning?"

In order to address these issues, this specification presents a reference architecture for SOA-based solutions. It provides a high level abstraction of a 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 a SOA. As stated above, underlying this layered architecture is a meta-model consisting of layers, capabilities, architectural building blocks (ABB), relations between capabilities, ABB's and layers, interactions patterns, options and architectural decisions. These will guide the architect in the creation and evaluation of the architecture. Note that per TOGAF, Capabilities are *ability that an organization, person, or system possesses to deliver a product or service.[add ref to TOGAF def]* Likewise, an architectural building block represents a basic element of reusable functionality, providing support for 1 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 reference architecture 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
   3.1. intermediary industry architectures
   3.2. concrete solution architectures
4. The SOA RA should promote and facilitate business to IT alignment.
5. It should address multiple stakeholder perspectives.
   5.1. For organizations implementing the RA within their enterprise:
      5.1.1. The reference architecture should be generic enough to be independent of vendor solutions.
      5.1.2. The reference architecture should define standard capabilities, building blocks, architectural decisions and other attributes to create a framework of understanding sufficient to enable an assessment of conformance.
   5.2. For Product Vendors:
      5.2.1. The reference architecture should provide a set of standards and enough specificity that they can use it to drive evaluation of compliance with those underlying standards.
   5.3. For Integrators:
      5.3.1. The integrator should be able to use it as a model to define specific constraints and directions for SOA implementations.
   5.4. For Standards Bodies:
      5.4.1. The RA should provide a reference against which they can extend standards, or provide guidelines, and more detailed levels of specificity etc.
6. The manner in which it is instantiated should be determined by the user of the RA.
7. It should use the fewest number of layers to depict the possible combinations and elements of a service-oriented architecture.

4 Basic Concepts

A number of key and fundamental concepts recur throughout the SOA RA. We will summarize them here:

Distinction between logical and physical/ design time and runtime elements of the SOA

1. All runtime elements that are part of the physical architectural are actually running in the operational systems layer.

2. 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 vertical / supporting layers of the SOA RA. The four horizontal layers deal with enabling the business capabilities required by the application running on the architecture. The four vertical 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.

3. 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 consumer 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.

4.1 Frequently Asked Questions

We would like to respond to some commonly asked questions in the early part of this document so as to level set the basic principles and set the tone for the rest of the document.

1. Question: Does the consumer layer consist primarily of things like GUI or is it a programmatic means of access to the services or both?

Answer: 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 graphical user interface or an API-like connection to the services.

2. Question: Can integration layer access services?

Answer: 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[2], be invoked directly by the consumer without the level of indirection associated with an Integration Layer.

3. Question: Which layer is responsible for invoking the service end point?
Answer: The consumer layer. 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.

4. Question: Who provisions the service?
Answer: Provisioning means making the service available for consumers to be able to invoke, then who does it? Governance has registry which needs the service meta data for the consumer to find and bind.

5. Question: Are all the architectural building blocks (ABB) 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?
Answer: not all architectural building blocks are required for every single implementation of an SOA. Instead every project will choose from among the building blocks within each layer of the SOA and select those that are appropriate for the particular project. In the case of use for an enterprise architecture standard within an organization, there will be patterns of architectural building blocks 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 architectural building blocks along with product selections and implementations.

6. Question: 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?

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

7. Question: Can you tell us about how the 9 layers would be serialized in practice?
Answer: The 4 cross cutting are basically supporting capabilities realized through vendor products. The 4 functional or horizontal layers will support the functional capabilities of architecture. The operational systems layer will provide the actual runtime for all layers, vertical or horizontal.

8. Question: So what are the architectural building blocks in each layer?
Answer: They are the key building blocks of that layer. You can think of them as the components providing key capabilities that are expected from that layer. For example, take the Integration Layer. You expect this layer to have some means of meditation, routing, protocol transformation. 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.

[2] Please refer to the Open Group Service Integration Maturity Model (OSIMM) for a discussion of the levels of maturity of an SOA.

9. Question: So what kind of building blocks do we have?
Answer" We have some 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: the services that are required to support the loan processing application
- Component layer: the software components that need to be built that support the realization and implementation of the services.
- Operational systems layer: the actual runtime environment in which the components, legacy systems, all backend applications, packaged applications reside and run.

Then we have the cross cutting layers which include the integration layer that serves as the point in which 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 pertaining to each of the layers, note that it is crosscutting 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 or actual data or analytics.

The quality of service layer ensures quality of service 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. This includes security, availability, all 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 relates to and touches upon all other functional and cross cutting layers of the foundation.

10. Question: Events are cross cutting, where do they reside and where are handled?
Answer: We have agreed to treat the cross cutting nature of events within the integration layer.

11. Question: Where is policy defined and where is it enforced?
Answer: Policy definition is the responsibility of the Governance Layer. The definition of policy enforcement points and policy enforcers is the responsibility of the individual horizontal, functional layers, such as Service, Business Process, etc. The enforcement of those policies is then the responsibly of a cross cutting layer, and we have chosen to consolidate that within the QoS Layer. 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.

12. Question: Where is business rule defined?

Answer: Business rules are 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. There business rule definition and management is the responsibility of the Governance Layer and its validation and enforcement in QoS Layer.

13. Question: Are ABBs and artifacts we create as part of methods the same?
Answer: Not in all cases. For example, A Process Description or a Process Model artifact is used by the Business Process ABB itself.

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

Answer:
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 may be reason to trigger events in Information Architecture Layer.

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

16. Question: If someone asserts they have a SOA that conforms to the SOA RA, what would have to be true for that statement to be true?

Answer:

Some ABB's in the SOA RA are foundation and may be essential or mandatory and others are advanced or optional and may be required higher maturity in SOA.

If we claim we have a SOA we should compare to the reference architecture and if there are not certain things there; logically (build your own or multi-vendor); should identify certain key building blocks. A set of capabilities at the logical level. The corresponding building blocks should be present. Overhead levied on projects with very small funding. Covers small to very large projects: what do they need to do to be compliant.

There are implications in terms of resource requirements. We will see in RFP's compliance with the SOA RA. There will be a spectrum of implementation ranging from point to point to physically

brokered via one or more ESBs. Customer's architectures can be different but conformant. It is also a quality and compliance checking mechanism.

17. Question: Is the SOA RA Normative? How can the SOA RA be used ?

Answer:

The SOA RA is intended as 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 consumers 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 Architectural Building Blocks provided by the SOA RA for conducting assessments, designing or implementating architectures.

Also the SOA RA combines two different types of organizations in one model description: a vendor may choose to use the standard in the development of SOA products and an organization that is in the process of assessing, designing and defining solutions using an SOA that leverages multiple products.

Prospective consumers may choose from alternatives that are specified within the SOA RA . Therefore they may choose to implement, for example, the integration layer of RA through a set of point to point communications between consumers and providers. Alternatively they may choose to conform to the capability outlined in the integration layer through the implementation of one or more enterprise service bus products or custom built implementations.

5 Overview of the Layers of the SOA Reference Architecture

The SOA Reference Architecture has nine layers which represent nine of the key clusters of considerations and responsibilities that arise in the process of designing an SOA solution or 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 2.1.1.

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 architectural building blocks) and physical (this aspect will be left to the implementation of the standard by an adaptor of the standard).

The requirements aspect reflects what the layer enables and includes all of its capabilities; the logical aspect includes all the architectural building blocks, design decisions, options, 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 will be left open to the implementer of the standard.

This specification provides specific focus on the logical aspects of the SOA Reference Architecture, 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 Components 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”, operational, or supplemental) nature (the Information Architecture, Quality of Service, Integration and Governance Layers). The SOA RA as a whole provides the framework for the support of all the elements of a SOA, including all the components that support services, and their interactions.

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

The SOA Reference Architecture enumerates the fundamental elements of a SOA solution or enterprise architecture standard for solutions and provides the architectural foundation for the solution.

As shown in Figure 1, the meta-model of the SOA Reference Architecture includes the following elements:

- *Layer*: An abstraction which contains a set of components such as ABBs, architectural decisions, interactions among components and interactions among layers, and supports a set of 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 (Architectural building block)*: A constituent of the architecture model that describes a single aspect of the overall model [12]. 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 architectural building blocks that recur consistently in order to solve certain classes of problems can be thought of as patterns of architectural building blocks. These patterns will consist not only on a static configuration which represents the cardinality of the relationship between building blocks, but also the valid interaction sequences between the architectural building blocks. In this reference architecture 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 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 instantiable solutions associated with them. An example of an option would be choosing SOAP or REST style SOA Services can be, since they are both viable options. Which option is selected 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 (Key Performance Indicators) and NFRs (Non Functional Requirements) to decide on standards and protocols to define a particular instance of an Architectural Building Block. This can be extended, based on the instantiation of the Reference Architecture to the configuration and usage of ABBs. Existing architectural decisions can also be reused 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 key performance indicator 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 attributes, (e.g. response time, etc.) and architectural cross-cutting concerns such as security.
- *Enabling Technology*: A technical realization of ABBs in a specific layer.
- *Information Model:* A structural model of the information associated with ABBs including data exchange between layers and external services. The information model includes the meta-data about the data being exchanged.

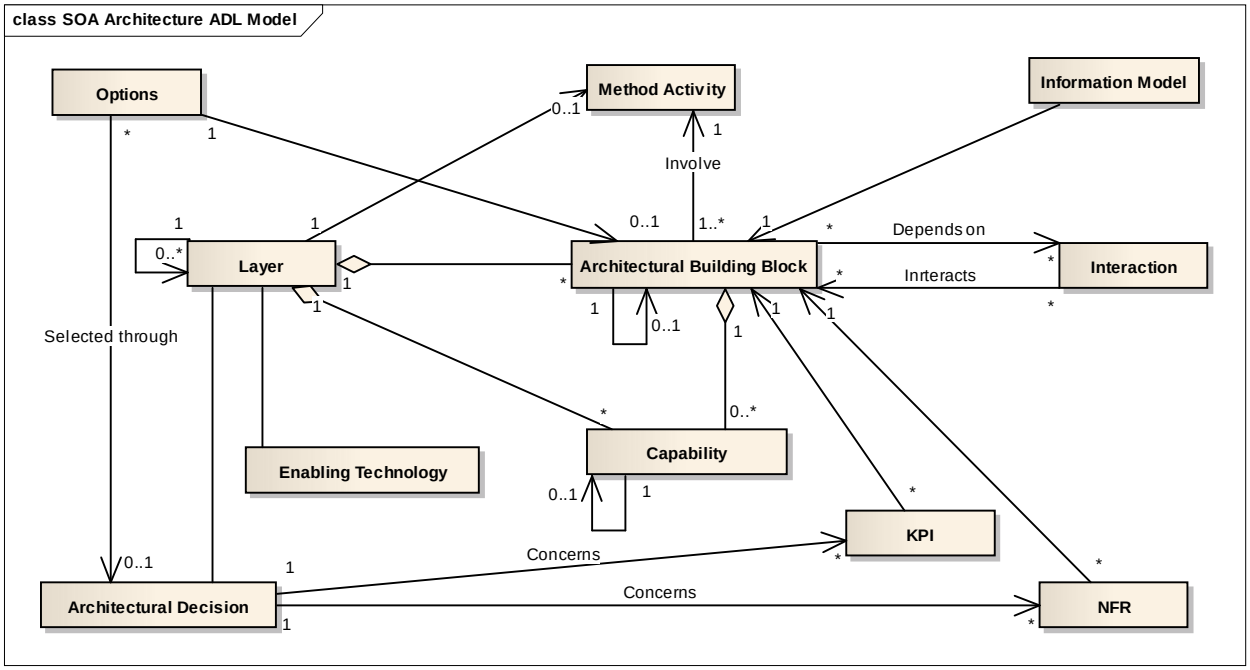


Figure 1 Meta-model for Instantiating the SOA Reference Architecture for a Given Solution

The following architectural diagram (Figure 2) depicts a 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 as a service or the service 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 Service 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 Reference Architecture represents SOA with a partially layered architecture. The degree to which a given organization realizes the full SOA Reference Architecture will differ according to the level of SOA maturity they exhibit, and the underlying requirements of the organization.

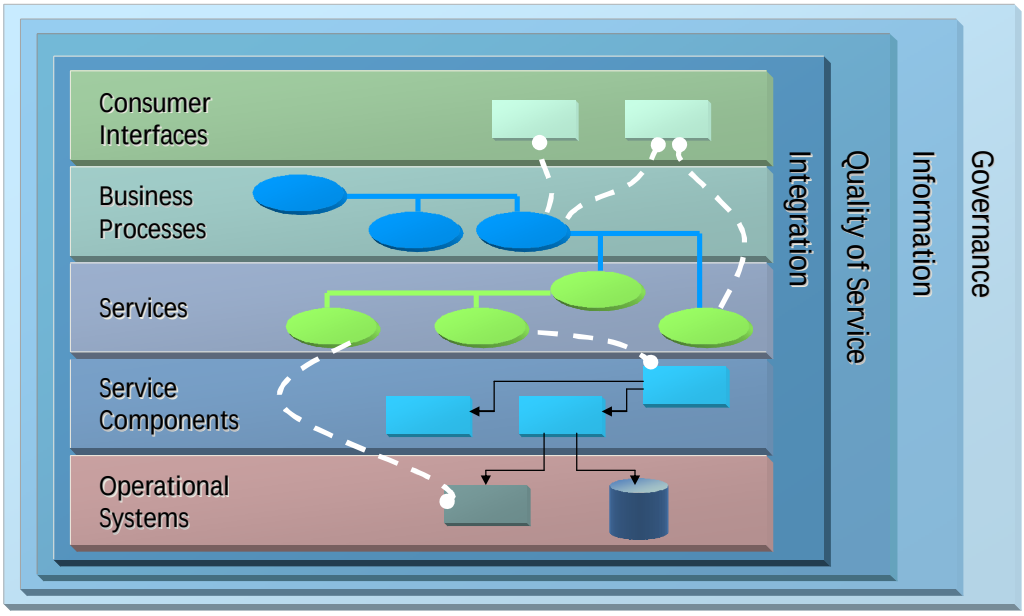


Figure 2 Logical Solution View of the SOA Reference Architecture

Figure 2 illustrates the multiple separations of concern in the 9 layers of this reference architecture. The SOA Reference Architecture 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. The need for both intra-and inter-enterprise SOA is important, with the role of SOA as the foundation of cloud computing and SaaS.

The main point of the provider/consumer separation is that there is value in decoupling one from the other along the lines of a 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. The lower layers

(Services, Service Components and Existing Application Assets) are concerns for the provider and the upper ones (Services, Business Processes and Consumers) are concerns for the consumer. Below we will describe each layer and in the subsequent sections, describe the relationships between the layers.

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

6 Capabilities and the SOA Reference Architecture

A capability, as defined by the Open Group is *"an ability that an organization, person, or system possesses" [TOGAF 9 XX]*. 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. For example, marketing, customer contact, or outbound telemarketing". In this document the term has been refined to a "Technical Capability", where a capability is the ability to service a technical requirement by the SOA. For example, "the ability to support service mediation" or "content-based routing" are examples of technical capabilities required by and supported by the 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 TOG 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 ABB's and Layers 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 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 meta-data 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 technical capabilities in turn map to capabilities and ABBs supported by actual SOA Reference Architecture layers.

A capabilities 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 RA helps us further, by determining if there are interdependencies 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 RA needs to satisfy.

Figure XX below shows how the relationships and Figure XX1 shows the derivation of SOA Reference Architecture Layers and dependent capabilities.

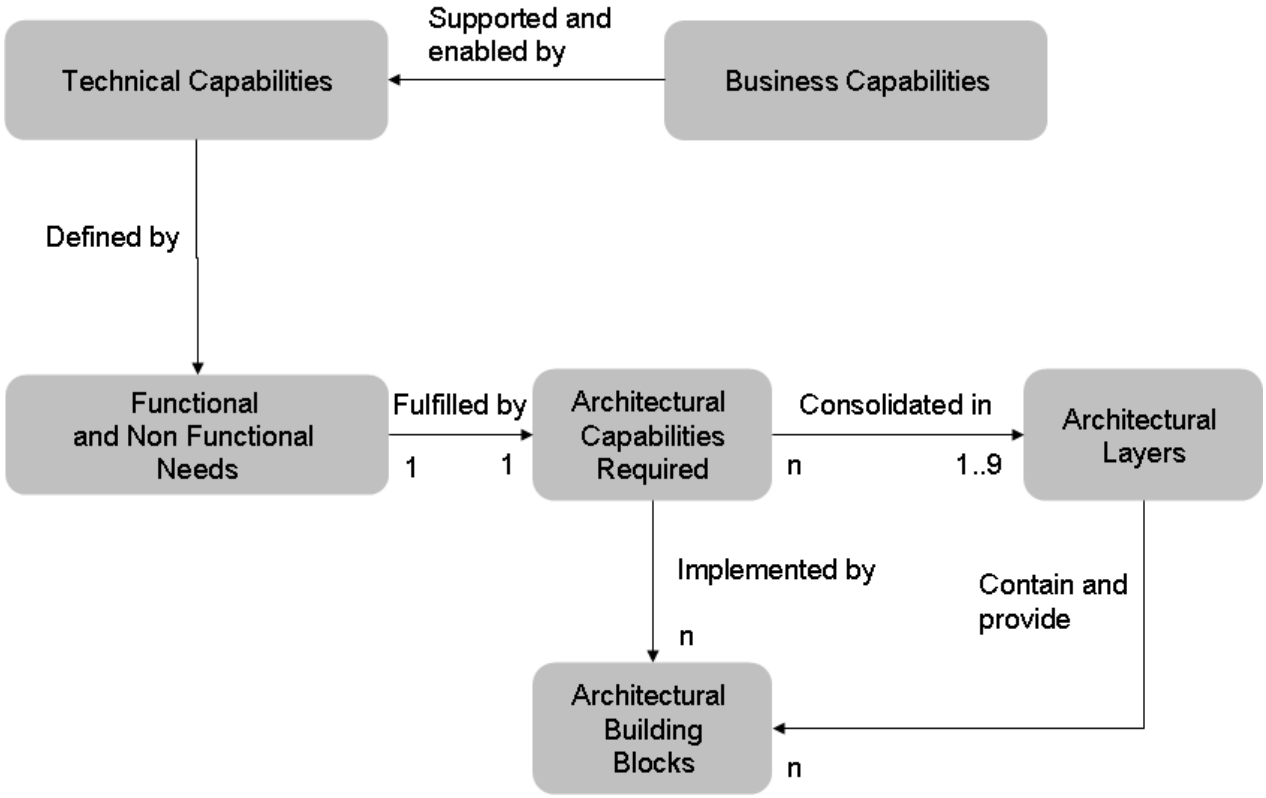


Figure 3 Relationships among Requirements, Capabilities, Building Blocks and Layers

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

7 Description of Layers

7.1 Assumptions

An SOA is defined by the set of functional and non-functional requirements 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. Non-functional service aspects 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 non functional requirements 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 fulfilment of any service requirement may be achieved through the capabilities of a combination of one or more layers in the SOA Reference Architecture.

Services themselves have a contract element and functional element. The contract defines what the service does for consumers, while the functional element implements what a service contracts to provide. 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 Reference Architecture is a key aspect in developing a service-oriented architecture for an enterprise [2][3].

In the description of layers, for each layer we provide:

1. 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.
   In general we follow a theme where each layer has a part which supports a set of capabilities/ ABB's which support the interaction of the layer with other elements in the 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 ABB's to support the layer's dynamic, runtime existence.
   Thus in this section we:
   a. Provide a structural overview of the layer, reviewing the different parts of the layer
   b. Provide a description of the layer, applying the meta-model
   c. Describe the ABBs within each part, applying the meta-model

Below we will show a scenario which describes a typical flow of execution through the various layers:

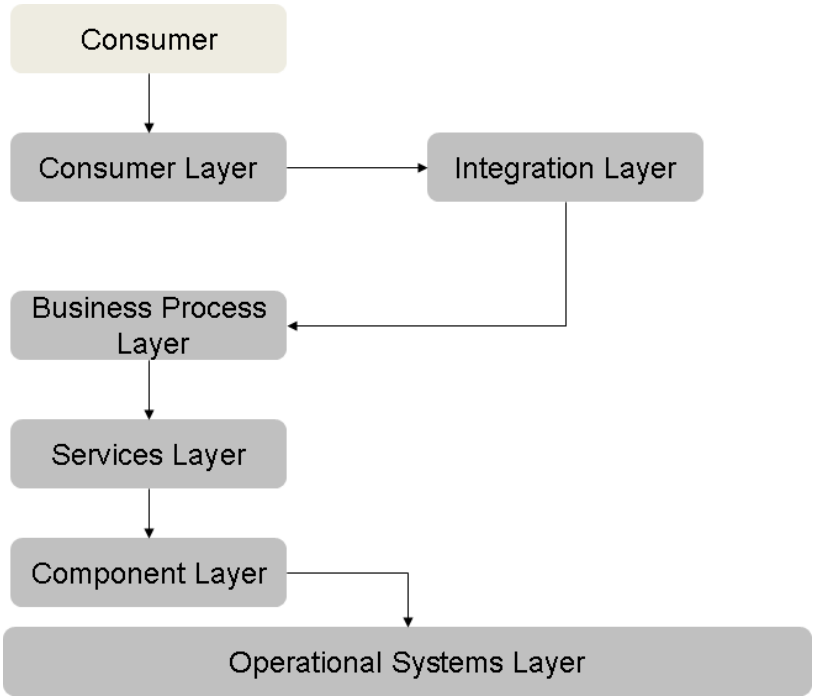


Figure 4 Typical Interactions among the Layers of SOA Reference Architecture

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 Business Process Layer which is using a Service.
3. It invokes the Service Layer.

4. The Services Layer discovers and invokes service components
5. Service components invoke solution components from the Operational Systems :ayer to carry out service time
6. The response turnover back up to the service consumer

8 Operational Systems Layer

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 services factors and governance).

8.1 Operational Systems Layer: Overview

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

1. All software and hardware infrastructure necessary to support the SOA and its components at runtime and design time (Tools).
2. All operational and runtime hosting of underlying physical system components
3. 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 "snap shot" and logical categorization and generalization of the run-time 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 architectural building blocks 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. These capabilities are outlined in the capability section.

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.

8.1.1 Context and Typical Flow

This layer supports multiple categories of capabilities for the SOA RA:

1. A set of capabilities to support the implementation of the functional element of the 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.
2. A set of capabilities for the runtime environment representing the runtime infrastructure of the SOA. This includes capabilities to support both the components required to support service functionality and those required to actually run the components and bulding blocks of the SOA RA itself. This includes capabilities for:
    a. The hardware, operating system components and the
    b. The solution building blocks, which are the runtime instances or realizations of the architectural building blocks (ABB's) of all layers in the RA that have been selected for inclusion in a particular operating environment.
3. A set of capabilities to support traditional infrastructure service support (storage, etc.) and virtualized services used from within the cloud.

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.

8.1.2 List of Capabilities

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

1. **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.

2. **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 RA itself.

3. **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 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 operational layer.

8.1.3 List of Architectural Building Blocks

The architectural building blocks responsible for providing these sets of capabilities in the Operational Systems Layer are:

#	Capability Category	ABB Name	Supported Capabilities
1.	**Service Delivery**	Solution Component	1 - 4
2.		Implementation Controller	5 - 10
3.		Integration Layer: Integration Controller	5
4.		Applications (Packaged and Custom)	2, 4
5.		Legacy System	4
6.		Database	3
7.		QoS Layer: Policy Enforcer	
8.		QoS Layer: Access Controller	2
9.	**Runtime Environment**	Runtime Hosting Environment	11 - 13
10.		Solution Platform	12
11.		Solution Building Block	15 - 16
12.		Deployment Unit	14
13.	**Virtualization and Infrastructure Services**	Hardware	17
14.		Virtualized Infrastructure	18 - 19
15.		QoS Layer: IT Systems Manager	20
16.		QoS Layer: Security Manager	21

Table 1 ABB to Capability Mapping for the Operational Systems Layer

8.2 Operational Systems Layer: Details of ABBs and Supported Capabilities

8.2.1 ABBs Details

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 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 a 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 meditating 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 QoS Layer: Policy Enforcer

See Policy Enforcer ABB in the QoS Layer.

8.2.1.8 QOS Layer: Access Controller

See Access Controller ABB in the QoS 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 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, an 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 QOS Layer: IT Systems Manager

See IT Systems Manager ABB in the QoS Layer.

8.2.1.16 QOS Layer: Security Manager

See Security Manager ABB in the QoS 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 supports:

1. solution components which provide the functional capability of services and the solution
2. run-time environment required by the SOA and that runs the actual solution components and their supporting infrastructure
3. supports the interfacing with underlying infrastructure, so that they can be virtualized and leveraged as such by the underlying architecture

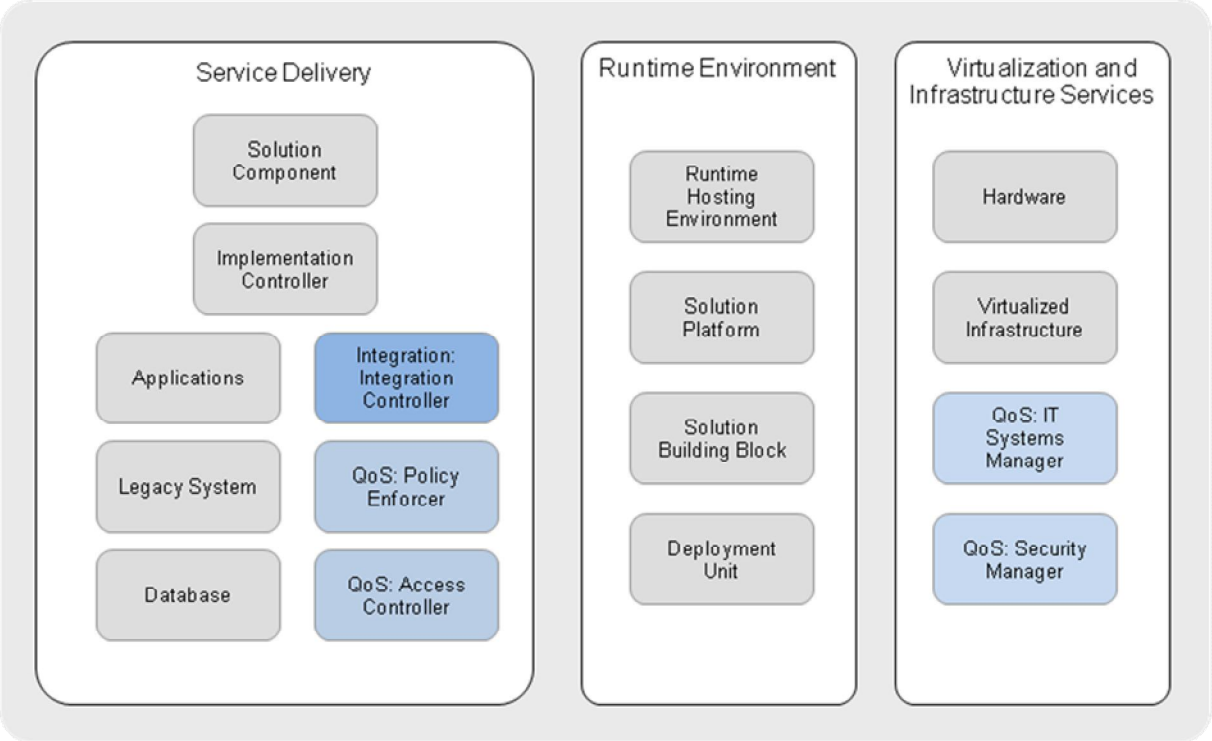


Figure 5 ABBs in the Operational Systems Layer

8.3 Operational Systems Layer: Interrelationships between the ABBs

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 define interfaces for the subsystems. Requests are first validated as secure by the Access Controller ABB and Security Manager ABB in the QoS 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 exposing the underlying infrastructure in an on-demand manner, encapsulating the invoking ORE and enabling rapid scaling.

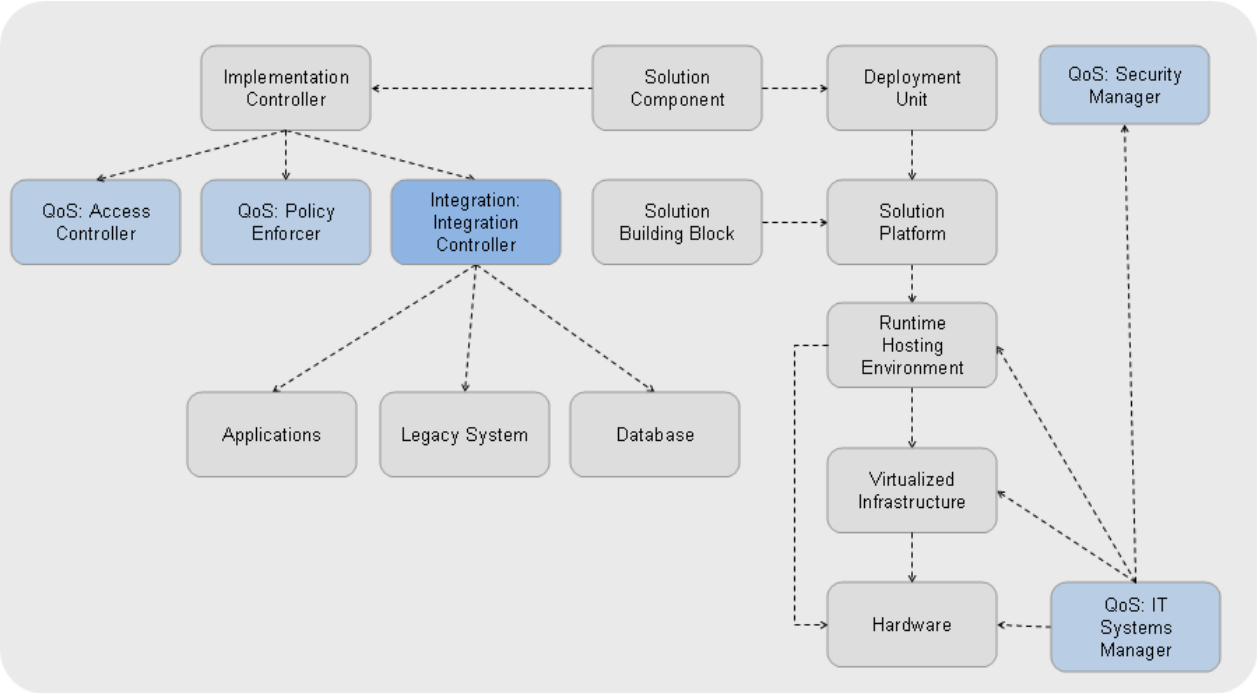


Figure 6 Relationships among ABBs in the Operational Systems Layer

8.4 Operational Systems Layer: 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 Reference Architecture including of course the cross-cutting and supporting layers, such as 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.

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.

8.4.1 Interaction with Cross Cutting Layers

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

1. It relies on QoS Layer for following capabilities:
    a. Ability to authenticate/authorize for service invocation.
    b. Ability to enforce operational policies.
    c. Ability to monitor health and well being of underlying infrastructure and solution and applications deployed on the infrastructure.
2. It relies on Information Architecture Layer for following capabilities:
    a. Ability to store and retrieve meta-data and data.
3. It relies on Integration Layer for following capabilities:
    a. Ability to invoke business processes and/or services.
    b. Ability to transform data from one format to another.

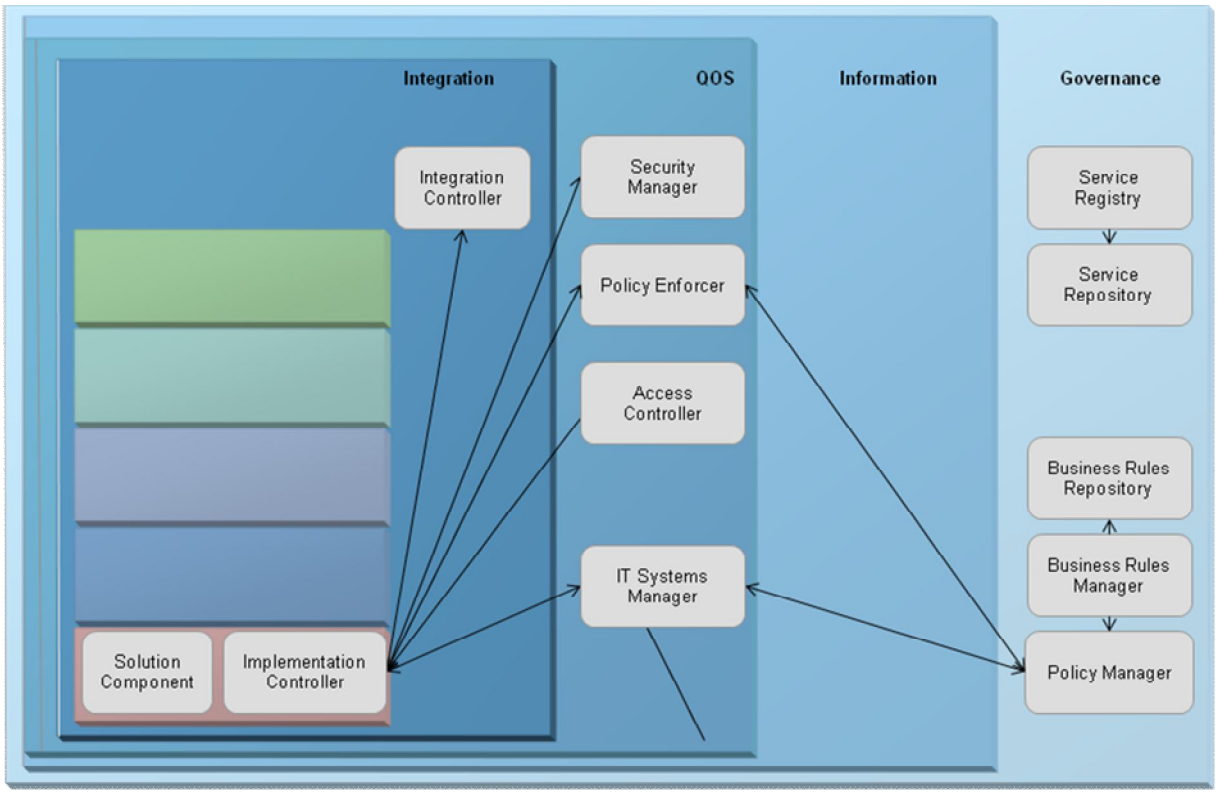


Figure 7 Key Interactions of Operational Systems Layer with the Cross Cutting Layers

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

a. 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.

b. It leverages Access Controller ABB in the QoS Layer to enforce access privileges and Policy Enforcer ABB in the QoS Layer to enforce policies. These ABBs from the Qos Layer enables the Operational Systems Layer to operate across platforms and support a consistent set of policies for specific scenarios, and limits the amount of associated risk. The Access Controller ABB and Policy Enforcer ABB in QoS provides a single policy enforcement point 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 QoS 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.

c. It leverages IT Systems Management related ABBs in the QoS 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.

d. 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.

8.4.2 Interaction with Other 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 gets rendered into Solution Component in the Operational Systems Layer during runtime and where as supporting ABBs get rendered into Solution Building Block in the Operational Systems Layers during runtime.

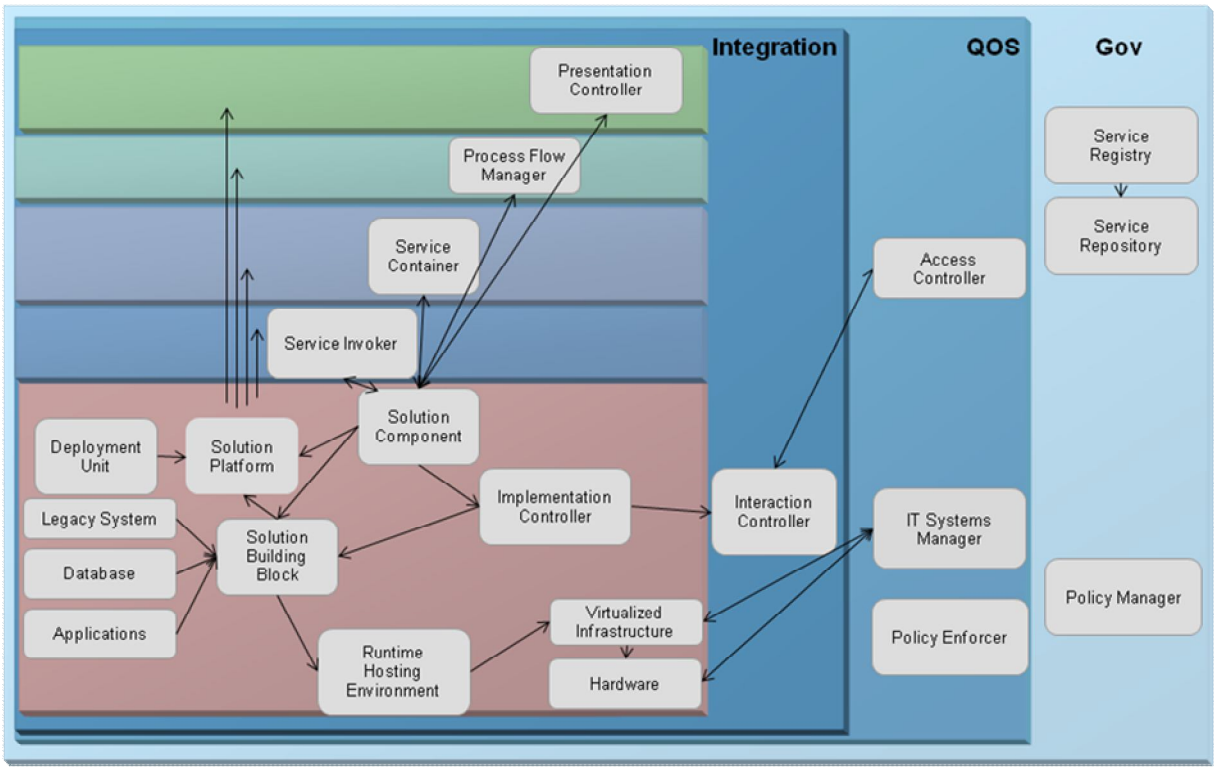


Figure 8 Key Interactions of Operational Systems Layer with the Other Horizontal Layers

8.5 Operational Systems Layer: Usage Implications and Guidance

The capabilities supported by the Operational Systems Layer including 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 a 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 reuse 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 reuse 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 [19]and .Net [20] applications
- Existing SOA services
- Legacy applications and systems
- Existing transaction processing systems
- Existing databases
- Existing package applications and solutions including ERP and CRM packages

8.5.1 Implementation Considerations

Considerations when using this layer include:

So what are the notes that a user of the SOA RA can take on using this particular layer:

1. When dealing with legacy, custom and COTS applications, plan on a layer to support composition/ decomposition and integrate with the underlying systems.

2. 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 an 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.
3. When dealing with virtualized infrastructure, it is important to consider the following:
    1. Isolation and containment services
        a. Multi-tenancy
        b. Data privacy (both data in motion and at rest)
        c. Auditability
        d. Authorization/ authentication/ access control
    2. Application support services
        a. Elasticity - Dynamic resource provisioning/de-provisioning
        b. Service location swareness
        c. Infrastructure QoS management (as opposed to Application Service QoS)
    3. Data integrity services
        a. Disaster recovery
        b. High availability clustering
        c. Data backup
        d. Data QOS management
        e. Data mobility
    4. Infrastructure accounting services
        a. Chargeback services
        b. Configuration management / auditability
        c. Capacity management services

8.5.2 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 Service 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 like ‘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 Block, as mentioned above.

In the figure below this deployment view of the SOA RA is visualized.

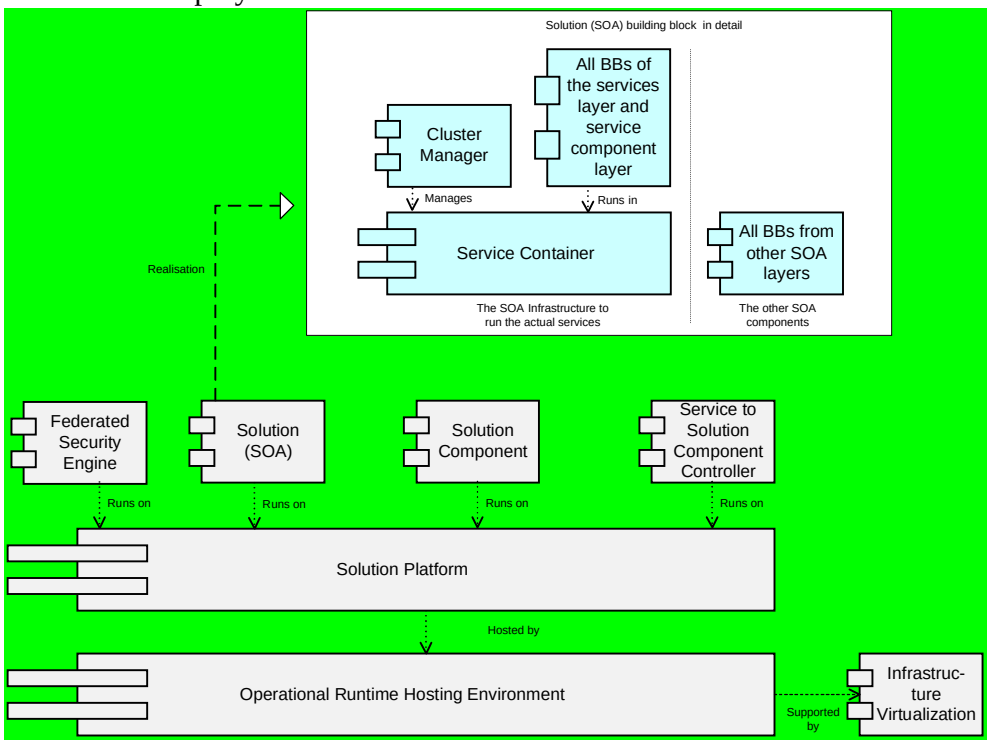


Figure 9 Deployment View of SOA Reference Architecture

The gray colored building blocks are from the Operational System Layer. The blue colored building blocks are partly generalized building blocks from the other layers.

9 Service Component Layer

9.1 Service Component Layer: 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 its functionality and its quality of service. 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 fulfills 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 like 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 they want/need to do
- 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:

1. 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.

2. 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).
3. Ability to publish and deploy the service component itself.
    a. Expose services in an interoperable manner
    b. Bind to the Operational Systems Layer at run time
    c. Publish service contract information in an interoperable and standards compliant manner so that other elements of the SOA can invoke it`
    d. Deploy the service into the associated "services container"

9.1.2 List of 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:

1. **Service Realization and Implementation**: This category of capabilities supports the realization of the services.
2. **Service Publication and Exposure**: This category of capabilities supports service exposure and service contract publication.
3. **Service Deployment**: This category of capabilities supports service deployment.
4. **Service Invocation**: This category of capabilities supports service invocation.
5. **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 *run-time* capabilities.

Design Time

Service Realization and Implementation

1. Ability to realize a service. For example, using component based design and development.

Service Publication and Exposure

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

Service Deployment

4. Ability to provides for the deployment of the physical service to the existing solution platform which contains the associated service solution component

Run Time

Service Invocation

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

Service Binding

6. Ability to support service interoperability
7. Ability to implements a part of the broker pattern
8. Ability to convert from the WSDL based service description by the WSDL stack on a platform to service calls supported by the platform. (in the case of a WSDL web service 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 List of Architectural Building Blocks

The architectural building blocks 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[3]	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
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.		QoS Layer: Policy Enforcer	11
13.		QoS 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 Service Component Layer: Details of ABBs and Supported Capabilities

9.2.1 ABBs Details

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.

[3] This can use any technology not necessarily SCA.

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 meta-data 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 component and associated functional and technical components. This includes service contracts and any other associated service meta-data.

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 run-time Service Components to a run-time 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 Service Layer. This includes invoking the Service Container to bind to and load the service component into the Service Container (a Service 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 meta-data from the Information Architecture 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 is housed.

9.2.1.12 QoS Layer: Policy Enforcer

See Policy Enforcer ABB in the QoS Layer.

9.2.1.13 QoS Layer: Access Controller

See Access Controller ABB in the QoS 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 <Buschman et al.>.

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

1. realization and implementation of services
2. exposure and contract publication of services
3. deployment of services
4. invocation of services
5. binding of services

The figure below is illustrating the ABBs supporting the capabilities of the Service Component Layer.

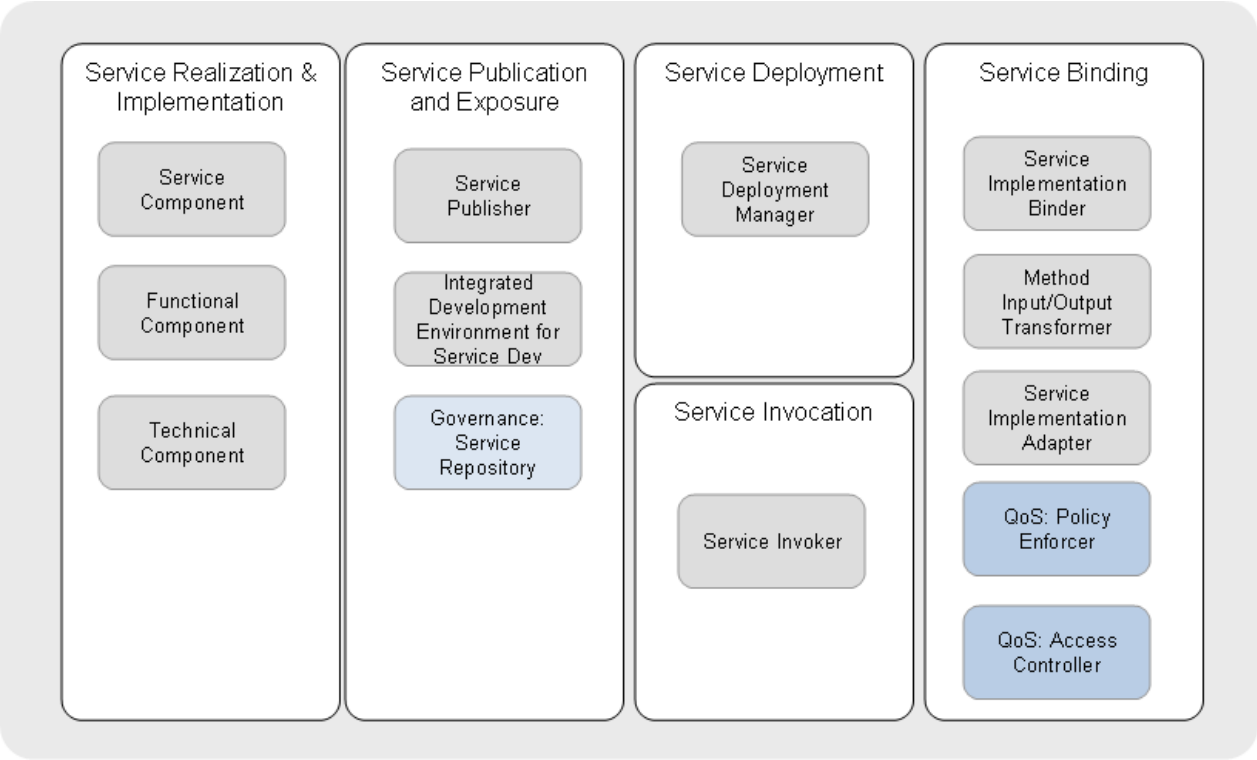


Figure 10 ABBs in Service Component Layer

9.3 Service Component Layer: Inter-relationships between the ABBs

As menitoned earlier the ABBs in the Service Component Layer can be grouped to groups:

1. ABBs that support the design time capabilities of the layer.
2. ABBs that support runtime capabilities of the layer.

The figure below illustrates the interaction flows among ABBs in the Service Component Layer enabling design time capabilities.

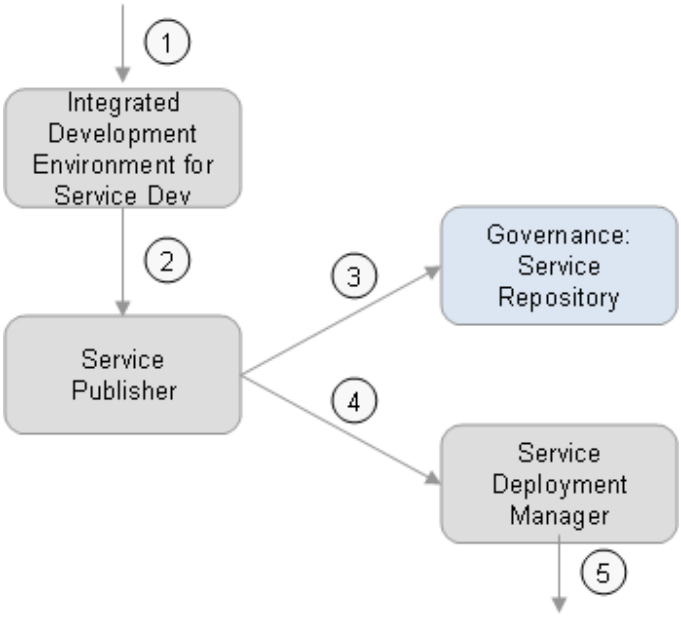


Figure 11 Illustrative Interaction Flow among Design Time ABBs in the Service Component Layer

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

1. During service design, and Integrated Development Environment (IDE) for Service Development is used to develop a service contract and specification using agreed upon standards
2. The service contract and specification is given to a Service Publisher ABB to publish the service.
3. 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
4. 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.

The figure below illustrates the interaction flows among ABBs in the Service Component Layer enabling runtime capabilities.

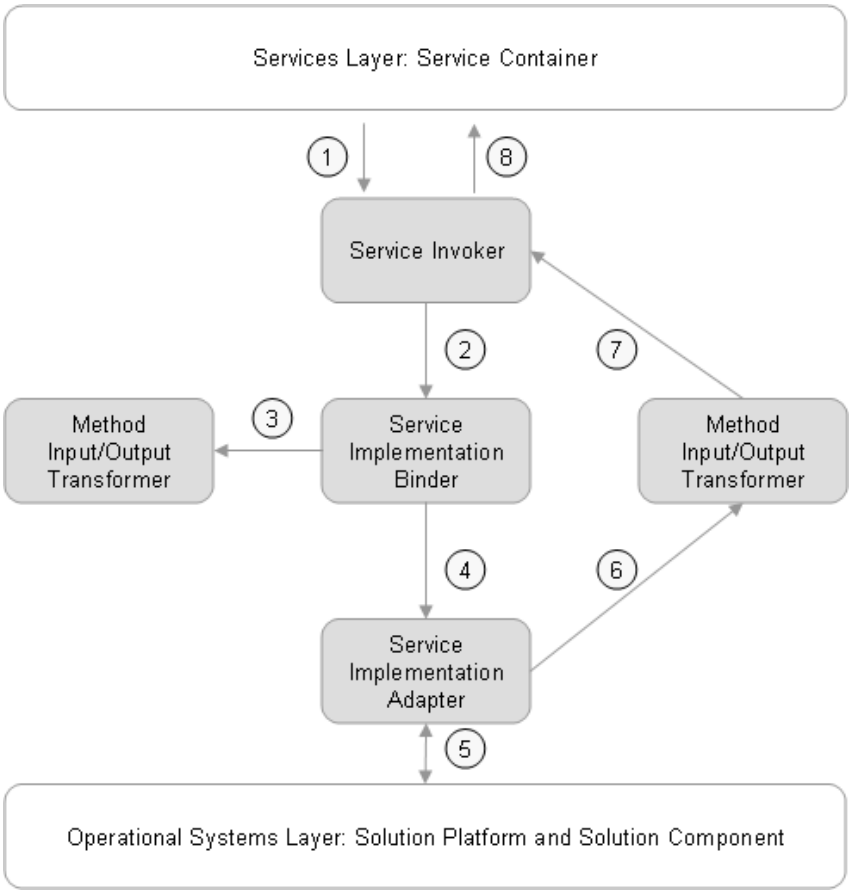


Figure 12 Illustrative Interaction Flow among Runtime ABBs in the Service Component Layer

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

1. Service Invoker ABB is invoked from all other layers of SOA RA (except Operational Systems Layer) and provides the ability for the Services Layer to invoke the Service Component realizing the services.
2. Service Invoker ABB calls the Service Component ABB which is the binding stack to bind to external layers.
3. 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.
4. Service Component ABB then pass control to the Service Implementation Adapter
5. The Service Implementation Adapter then maps the invocation into the Operational Systems Layer.

9.4 Service Component Layer: 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 quality of service and service level agreements)
- 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.

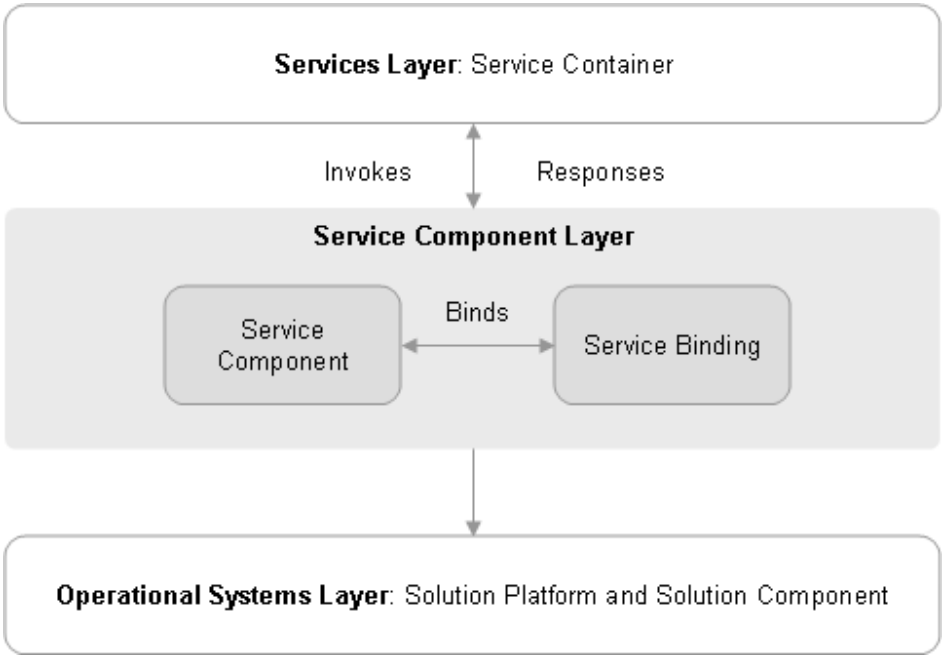


Figure 13 High Level Interaction of the Service Component Layer with Layers above and below in the SOA RA

The Solution Component ABB in the Operational Systems Layer can be thought 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. Solution Component ABB is 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 fulfil its responsibilities.

1. It relies on Governance Layer for following capabilities:
    a. Ability to store meta-data about services during design time.
    b. Ability to define and manage (storage, retrieval, etc.) of rules used by the components realizing the services.
2. It relies on QoS Layer for following capabilities:
    a. Ability to authorize during invocations on the underlying components.
3. It relies on Information Architecture Layer for following capabilities:
    a. Ability to store and retrieve meta-data and data required by the components.
4. It relies on Integration Layer for following capabilities:

Ability to transform data from one format to another

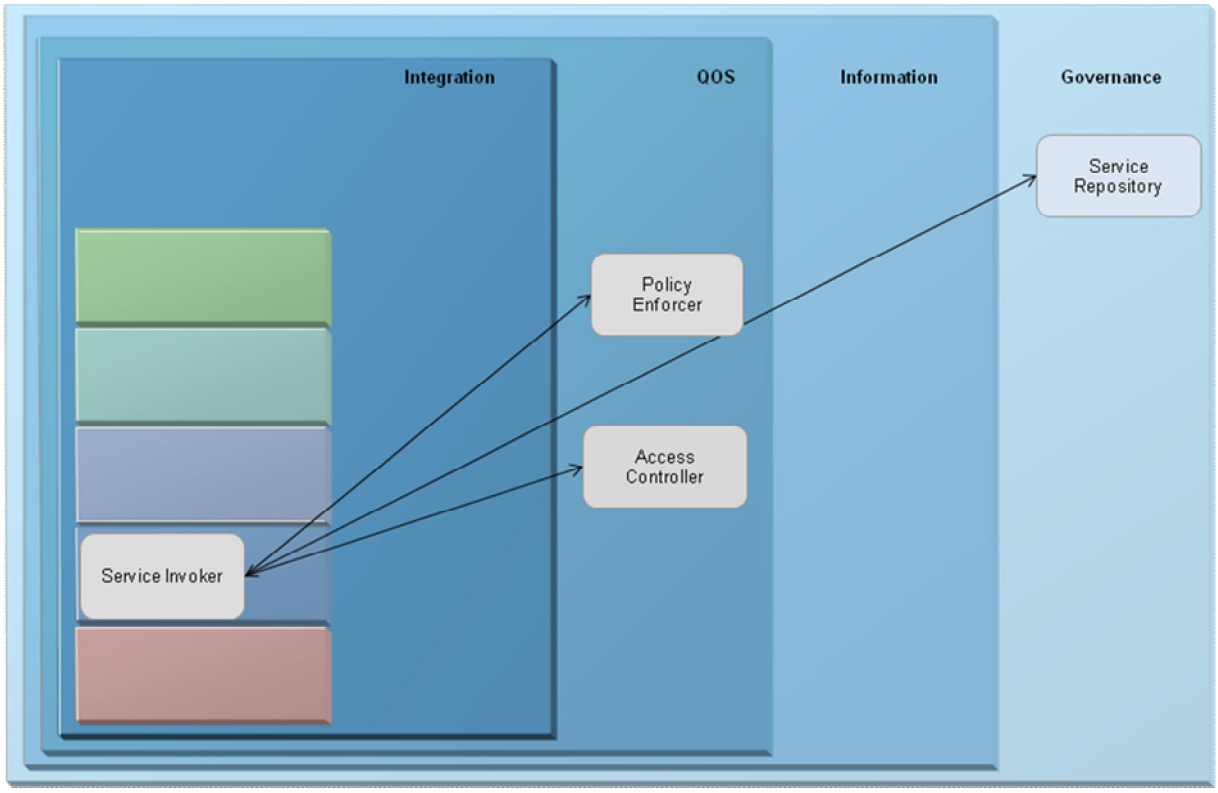


Figure 14 Key Interactions of Service Component Layer with the Cross Cutting Layers

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

a. It leverages Access Controller ABB and Policy Enforcer ABBs in the QoS Layer to enforce access control privileges and other policies.
b. It leverages Data Aggregator ABB, Data Federator ABB, Data Consolidator ABB, Information Meta-data Manager ABB and Data Repository ABB from Information Architecture Layer.
c. It leverages Message Transformer ABB and Data Transformer ABB from the Integration Layer to transform data from one format to another. Method Input Output Transformer ABB leverages these ABBs from the Integration Layer.
d. It leverages the Service Repository ABB in the Governance Layer to store meta-data 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 behaviours 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 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 Other Horizontal Layers

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

e. Service Invoker ABB interacts with the Service Layer and Integration Layer,
f. Service Publisher ABB interacts with the Service layer.
g. Service Deployment Manager ABB interacts with the Operational Systems Layer.
h. Service Implementation Adapter ABB interacts with the Operational Systems Layer.

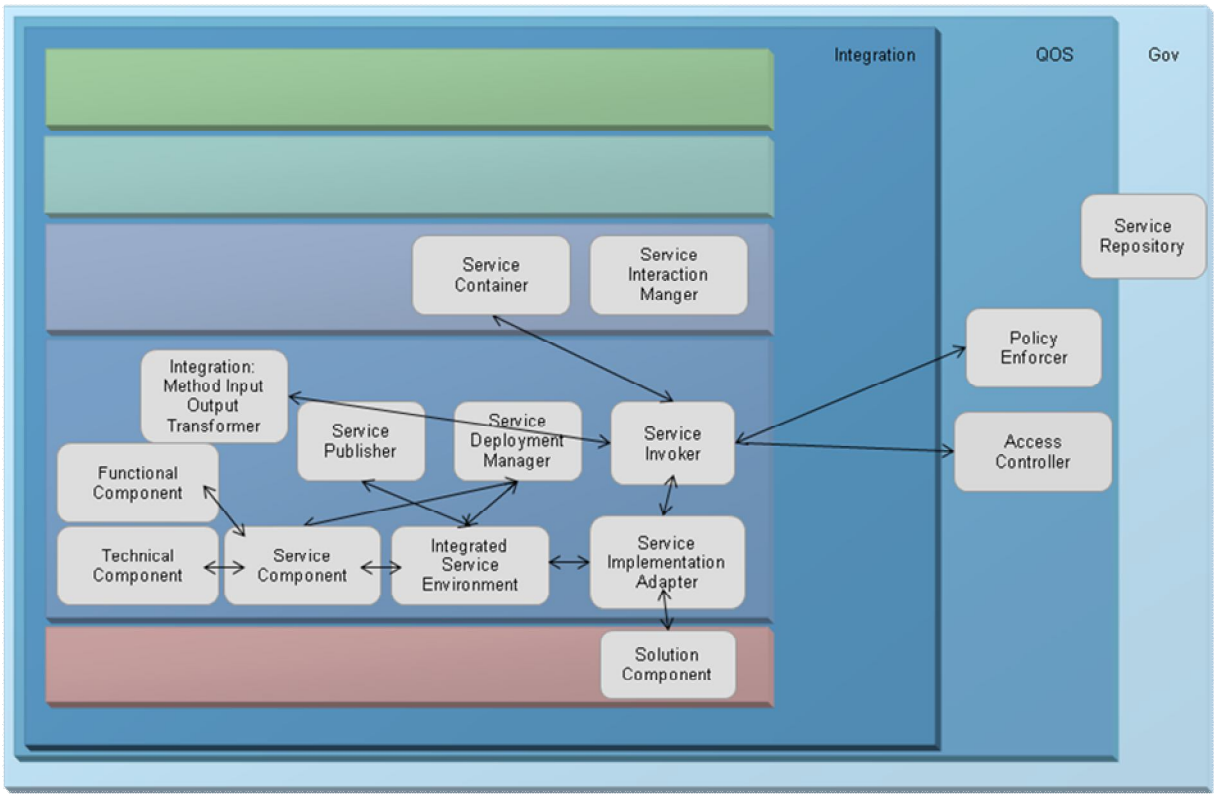


Figure 15 Key Interactions of Service Component Layer with the Other Horizontal Layers

Interaction with the Service Layer

The Relationship Between the Service Component and Services Layer

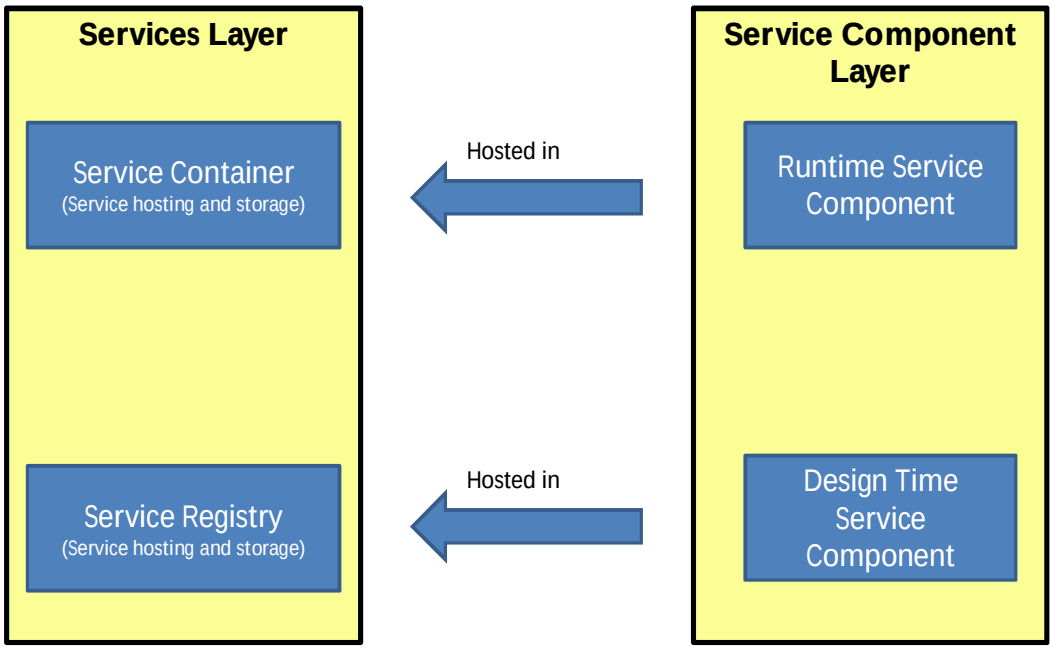


Figure 16 Relationships between Services Layer and Service Component Layer

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 will also be retired[4]. 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 Service Layer is illustrated in the following figure. Services are deployed in the services container in the Service 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

[4] 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.

service component in the Services Component Layer which is then bound to the solution platform and invoked in the Operational Systems Layer.

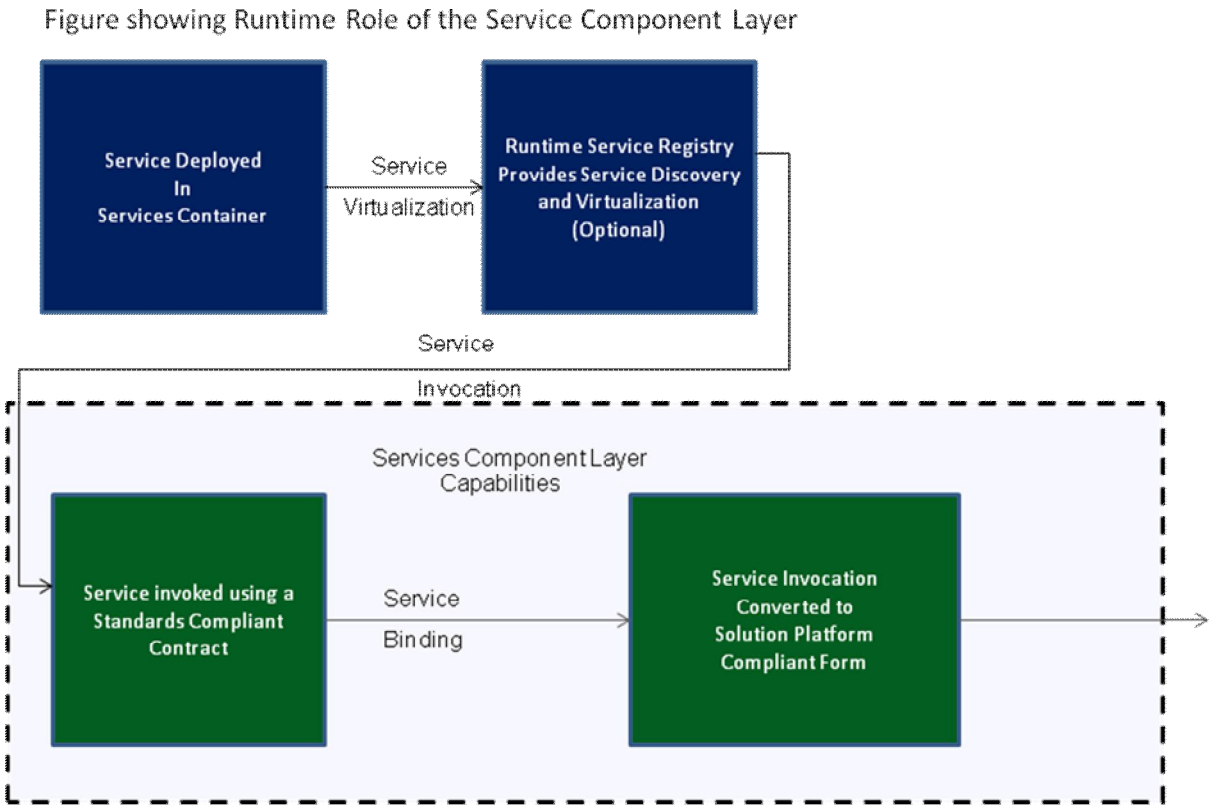


Figure 17 Showing the use of runtime capabilities

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 Service Component layer and the Operational Systems layer is an important element of an SOA.

Also, it is often the case that Operational Systems layer behaviors 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 existing Operational Systems Layer.

9.5 Service Component Layer: 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 inline with the technology roadmap of the organization.
- Manageability: whether the technology allows for the effective management of service components with respect to the 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 or is which represent the crosscutting concerns that are enablers for the functional layers, decisions regarding those crosscutting 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 particular application?
- What is an appropriate runtime environment for particular subset of an application?
- What kind of runtime capabilities are required in terms of nonfunctional requirements?
- Should integration always occur through an 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<- ->EJB or EJB<- ->Service)
- Where are transformations performed? In 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 concerns in 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 time frame. 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 time frame. Security refers to the ability of a service that provides 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 in **Error! Reference source not found.**where "*Service A*" is implemented using a combination of behaviour 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 behaviour 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"*.

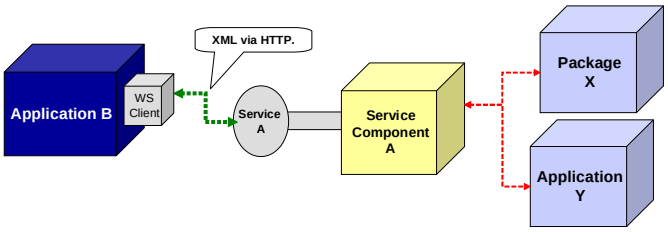


Figure 18 Service Components as a Facade

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 below 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 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.

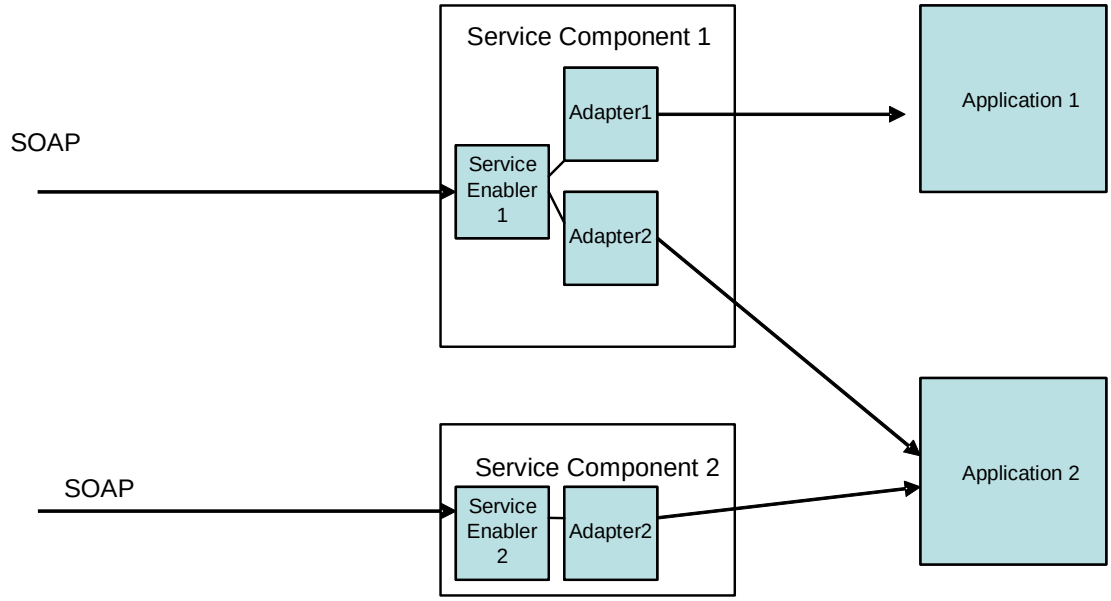


Figure 19 Interaction Flow in a Composition Scenario

As shown in figure above, there is a need of having 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 who 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 the figure above. 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 reused in both *"Service Component 1"* and *"Service Component 2"*.

10 Layer 3: Services Layer

10.1 Service Layer: 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 the 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. 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" was only existed in the programming model such as Java, 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. But it defines runtime capabilities for service deployment but the runtime instantiation of the architectural building blocks 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 [4]. 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 service 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:

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

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

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

10.1.2 List of Capabilities

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

1. **Service Definition:** This category of capabilities provides ability to define the service description.
2. **Service Runtime Enablement:** This category of capabilities provides ability to support service versioning, to support service binding decoupling a service from it implementation and provides ability to provision services.
3. **Policy Management:** This category of capabilities provides ability to manage and enforce policies associated with services.
4. **Access Control:** This category of capabilities provides ability to manage access to services.
5. **Service Clustering:** This category of capabilities provides 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 heart beat of the services.

Policy Management

8. Ability to support the integration of the QoS policy descriptions for services with the runtime elements of the Governance and the QoS 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 quality of service 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 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 quality of service (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 List of Architectural Building Blocks

The architectural building blocks 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 - 5
4.		Service Interaction Manager	6
5.		Governance Layer: Service Registry	3
6.		QoS Layer: Status Manager	7
7.	**Policy Management**	Governance Layer: Policy Manager	8 - 9
8.		QoS Layer: Policy Enforcer	10 - 11
9.	**Access Control**	QoS Layer: Access Controller	12 - 13
10.	**Service Clustering**	Cluster Manager	14 - 15

Table 3 ABB to Capability Mapping for the Services Layer

The architectural building blocks responsible for providing these sets of capabilities in the Services Layer are:

10.2 Service Layer: Details of ABBs and Supported Capabilities

10.2.1 ABBs Details

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 Service Repository ABB in Governance Layer during design time for search and reuse and Service Registry ABB in Governance Layer during runtime for service virtualizaton. A service is typically represented in a standard description language (e.g., Web Services Description Language (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 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/ Gateway

This ABB acts as a container or gateway by providing the environment 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 endpoints), 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 ABB's required the ability to support runtime invocation and running of services and integration with cross-cutting layers.

Within the Service Container are ABB's which enable it to invoke and execute service components, and support the integration with the cross-cutting layers – the QoS Layer, Integration Layer and Governance Layer. It leverages Service Repository ABB and Service Registry ABB in 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 ABB's 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 QoS Layer: Status Manager

See Status Manager ABB in the QoS Layer.

10.2.1.7 Governance Layer: Policy Manager

See Policy Manager ABB in the Governance Layer.

10.2.1.8 QoS Layer: Policy Enforcer

See Policy Enforcer ABB in the QoS Layer.

10.2.1.9 QoS Layer: Access Controller

See Access Controller ABB in the QoS 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 Services 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:

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

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

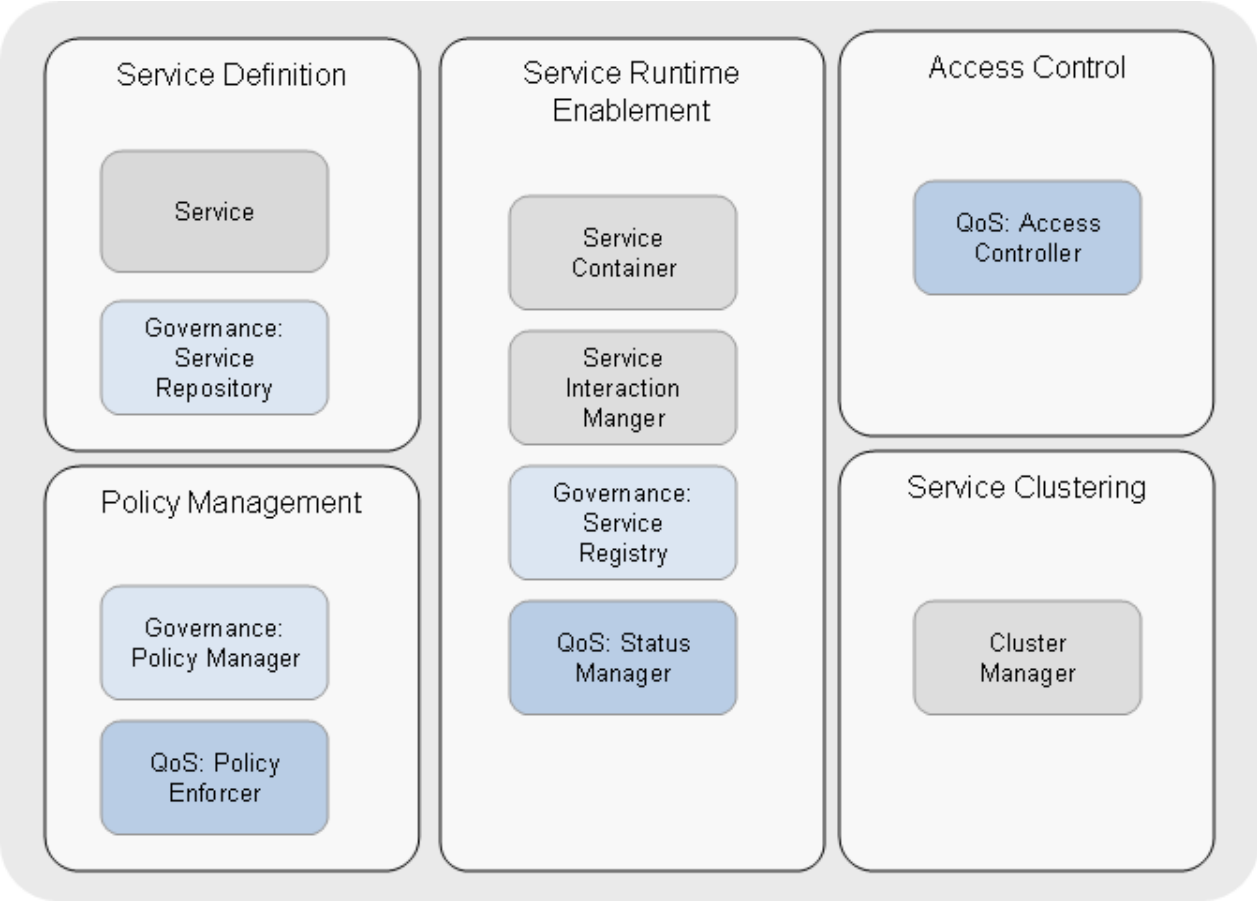


Figure 20 ABBs in the Services Layer

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

1. ABBs supporting design time needs are:
    a. Service ABB
    b. Service Repository ABB in the Governance Layer
    c. Policy Manager ABB in the Governance Layer
2. ABBs supporting runtime environment for the services are:
    a. Service Container ABB
    b. Service Interaction Manager ABB
    c. Service Registry ABB in the Governance Layer
    d. Policy Enforcer ABB in the Governance Layer
    e. Access Controller ABB in the QoS Layer
    f. Cluster Manager ABB
    g. Status Manager ABB in the QoS Layer

10.3 Service Layer: Interrelationships between the ABBs of the Services Layer

The figure below shows the interdependencies between the ABBs during design time and runtime.

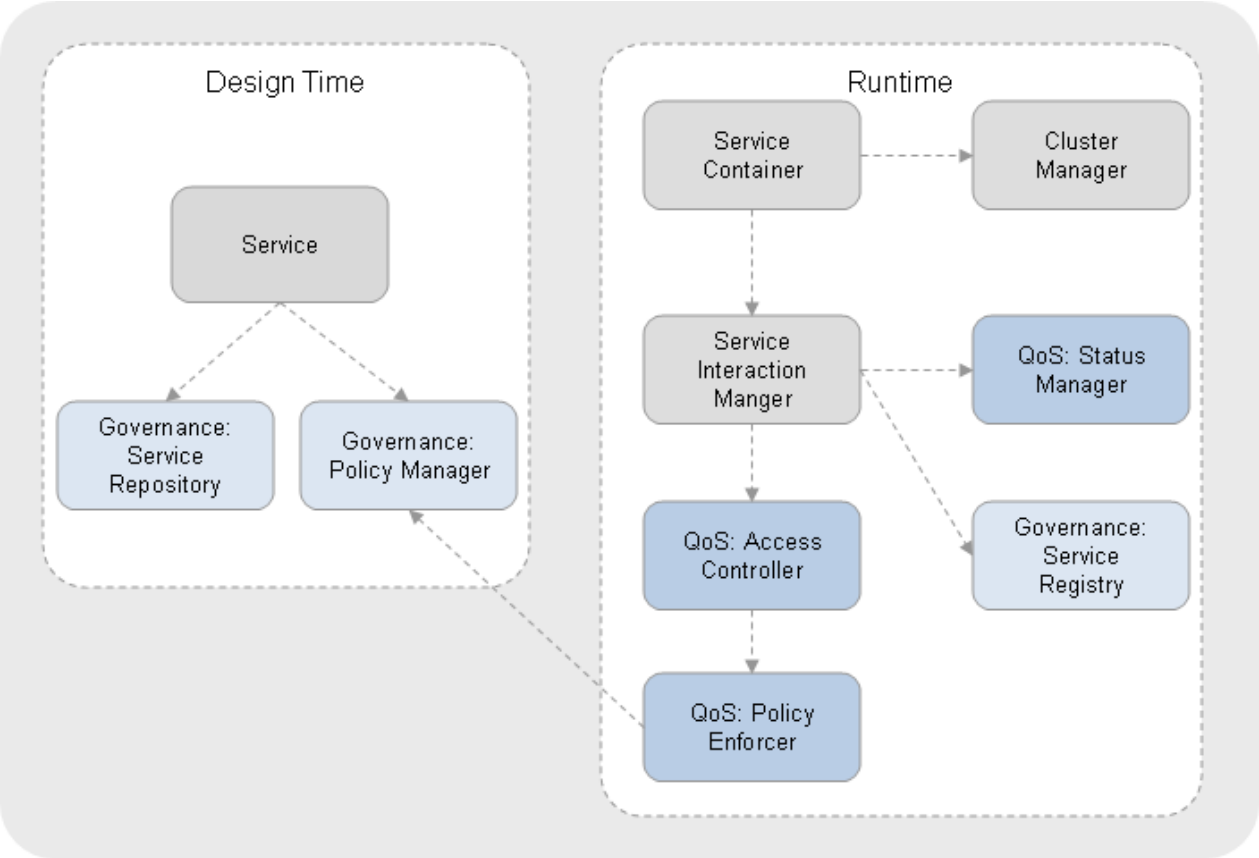


Figure 21 Relationships among ABBs in the Services Layer

During design time, information such as metadata about service contracts gets stored in the in Service Repository ABB in the Governance Layer and policy associated with services are defined using 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. Policy Enforcer ABB in the QoS Layer enforces service policies (including both quality of service (QoS) and security policies). Service Container ABB invokes the Service Component the Service Component Layer 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 Endpoint, 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.

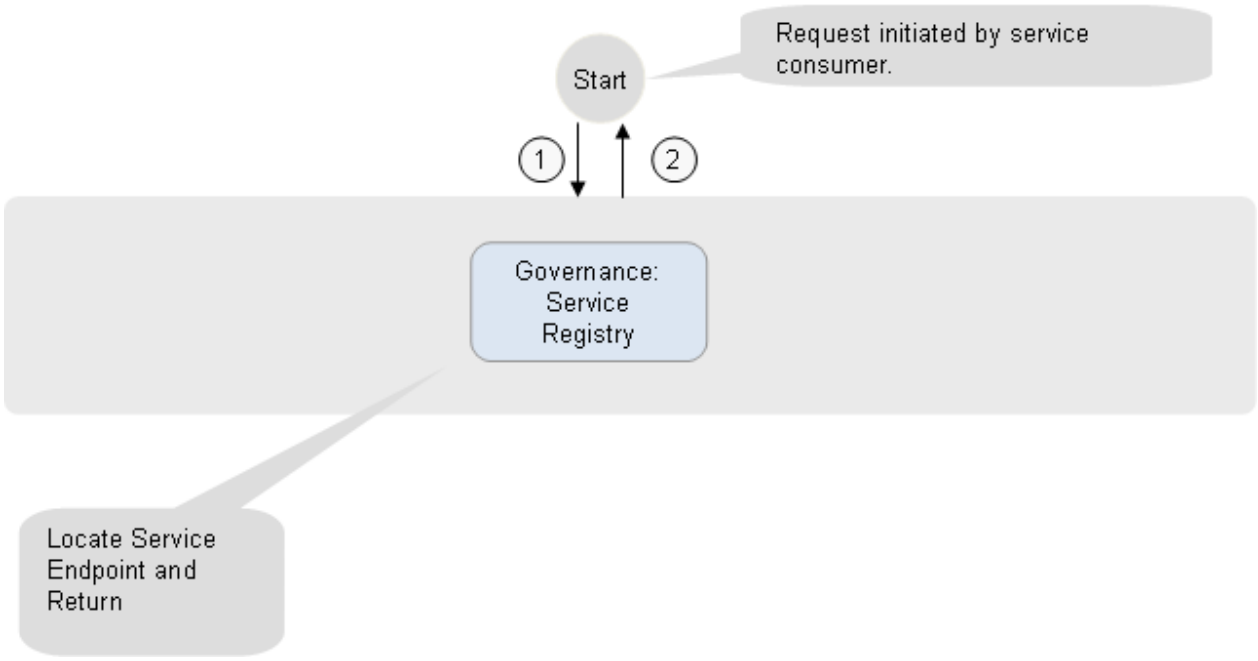


Figure 22 Interaction Flow for Service Discovery and Location

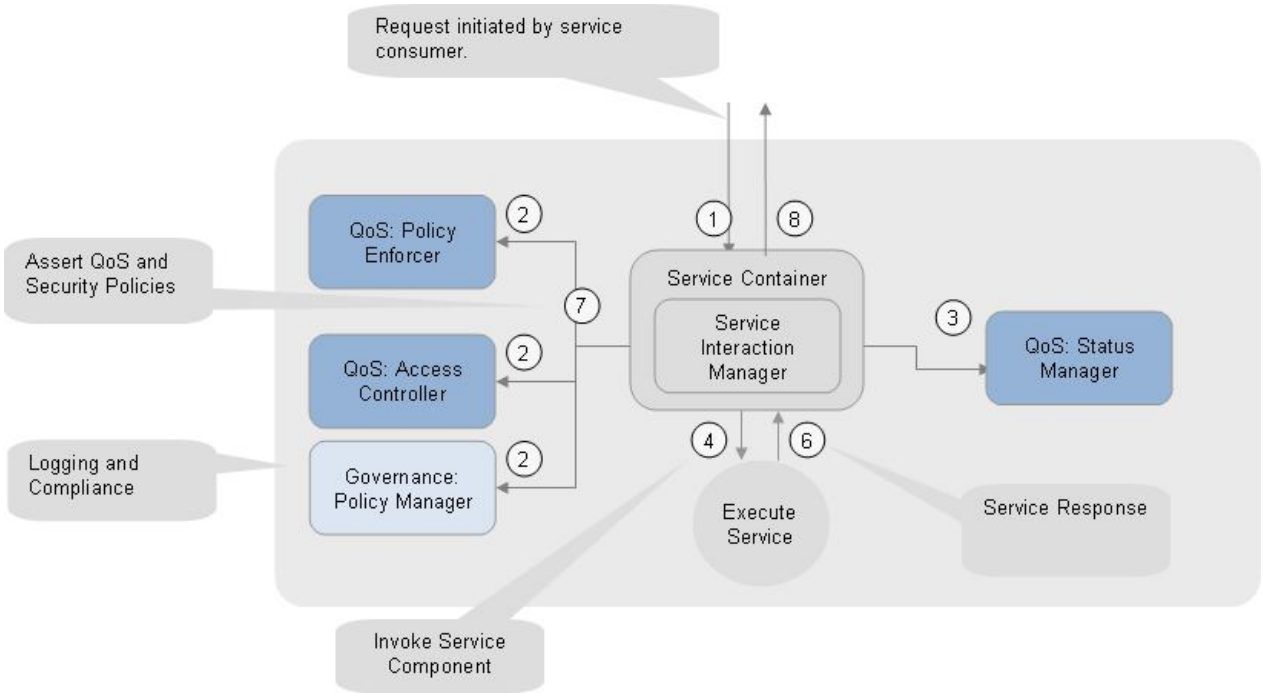


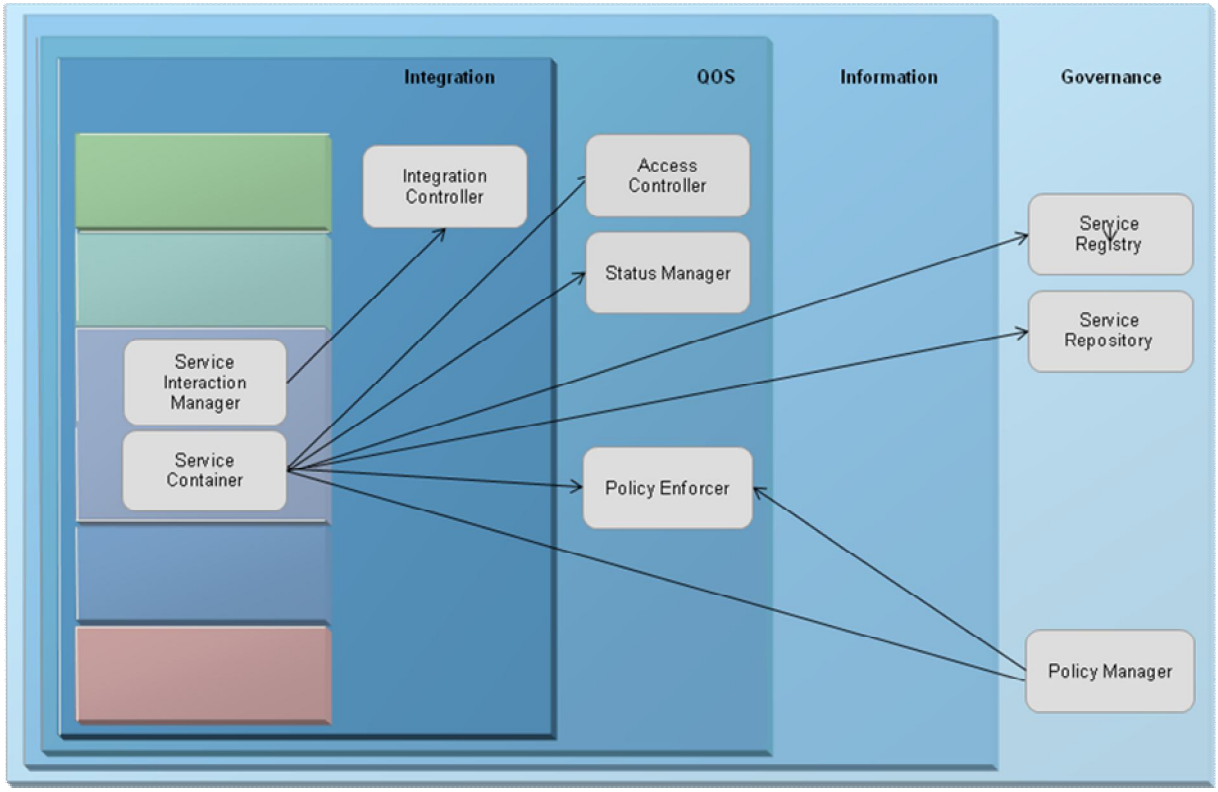
Figure 23 Interaction Flow for Service Invocation

10.4 Services Layer: 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 fine information needed to support the service, like policies and binding information. This relationship at runtime with enables late binding of services.

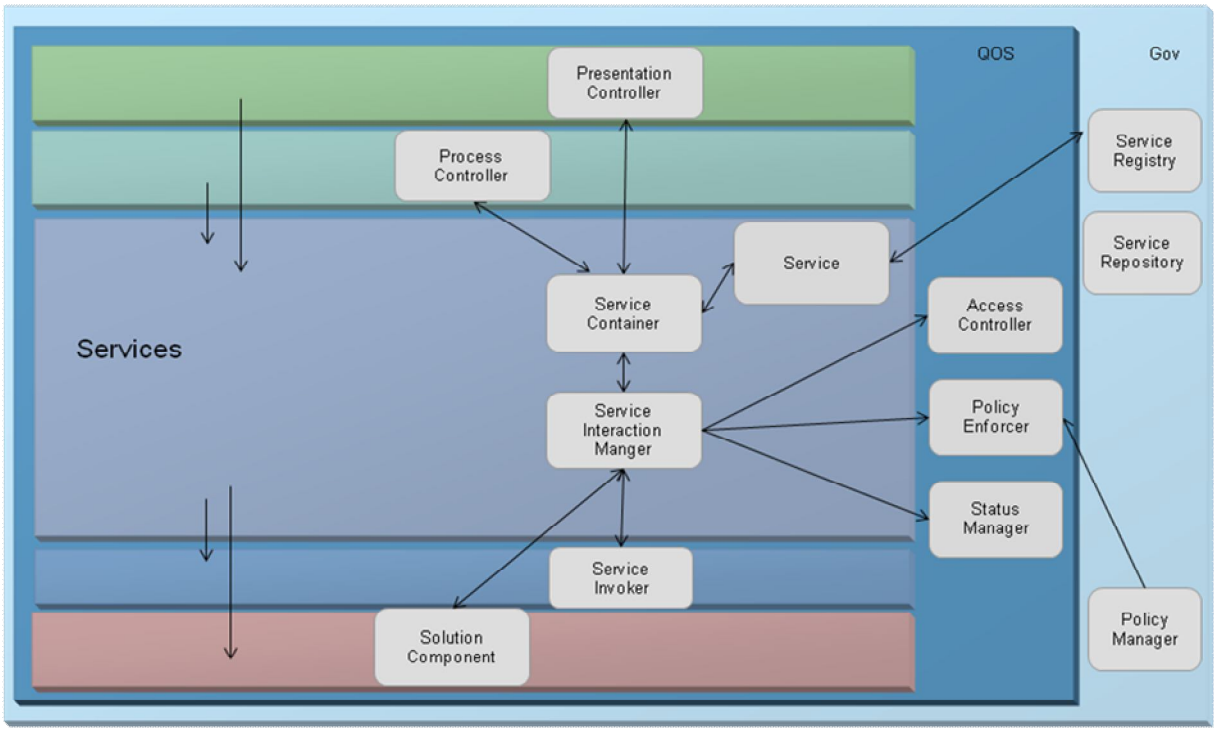
The Policy Manager ABB in Governance Layer, Access Controller ABB in the QoS Layer and Policy Enforcer ABB in the QoS Layer exchange and enforce policies ensuring standards compliant interaction that adheres to the governance regimen. The figure below illustrates these relationships.



The Service Container ABB also leverages Policy Enforcer ABB to enforce the service policies in order to deal with compliance of service level agreements of services.

10.4.2 Interaction with Other Horizontal Layers

The Service Registry ABB in the Governance Layer is where service consumers interact with the Services Layer to find the service endpoint, by which a service is invoked (the actual specification of what is a service endpoint 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 QoS Layer to assert the Quality of Service 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 is the QoS layer of the change. Likewise, the Status manager can interact with the Service interaction manager to change the status of the service.



To summarize:

The service repository provides design time storage of meta-data for services.

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

The service integration manager 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 uses the service policy and access controllers 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 and propagate it back via the Serivce 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 ABB's which define security policies in the Quality of Service Layer. In practice, it too acts as a Policy Enforcer.

10.5 Service Layer: Service Types and a Middleware View of the SOA Reference Architecture

The services layer includes aspects of the integration, information, and quality of service and governance layers. It includes the services that will be delivered by the given architecture, including both composite and atomic services. Service Categorization, the partitioning of services into groups is a characteristic of the Services in an SOA. It has practical implications in terms of the manner in which the business views and understands the SOA, the manner in which services are grouped into Services portfolios and built and created.

The figure below is the result of zooming into the services layer. It shows an underlying middleware and infrastructure service categorization of the services layer.

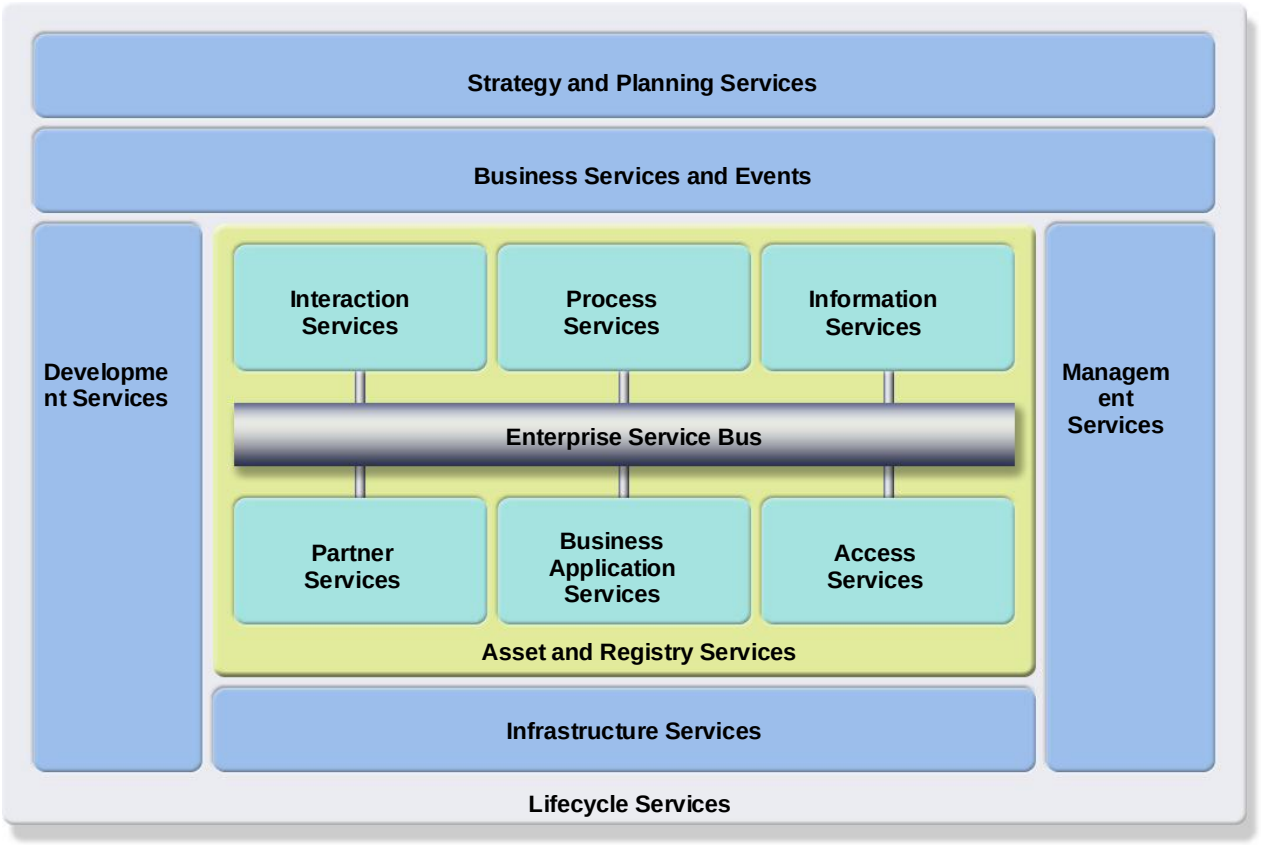


Figure XX: The Middleware View of the SOA Reference Architecture

10.6 Service Layer: 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 service 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 out 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 in 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 modelling to identify, specify and realize services, components and flows [1]. 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), Voice XML [18] (for IVR), and XML string (for XML client) can be done via XSLT [21] 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 a SOA solution at a logical level, the SOA Reference Architecture 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 Reference Architecture provides a well-factored decomposition of the SOA problem space, which allows architects to focus on those parts of a 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 a 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 architectural building block. Using the same SOA Reference Architecture, SOA business services can be delivered based on the same deployment framework.

11 Layer 4: Business Process Layer

11.1 Business Process Layer: Overview

11.1.1 Context and Typical Flow

With the advent of SOA came the promise of agility and flexibility. For an organization that has embarked on the journey of SOA would be successful in delivering the promise of agility and flexibility only when its the 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 past, business process flows were typically embedded in software components and user interfaces. The 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 in the reference architecture. The layer is called Business Process Layer and allows externalization of the business process flow in a separate layer in the architecture and thus having a better chance to rapidly change as the market condition changes.

The Business Process Layer covers process representation, 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 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 application or at best, a hardwired workflow. With service-orientation, a process can be realized by service compositions (which may be interim be 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 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 X 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.

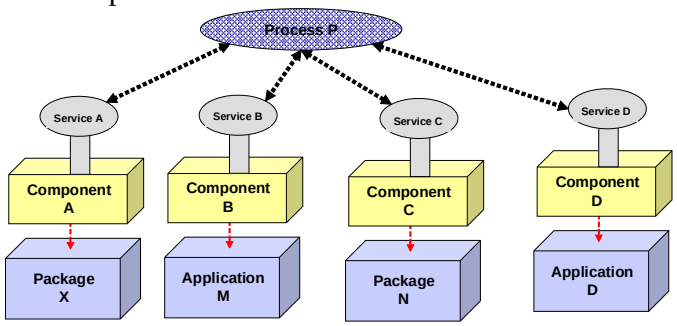


Figure 4: Services Orchestration

The Business Process Layer leverages the Service 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 Service Layer. In other words, the layer provides 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 allow 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 reusable services that can be modeled, analyzed, and optimized based on business requirements such as QoS (historical data described in the QoS 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 pin points by reusing services assets.

From interaction perspective, the Business Process Layer communicates with the Consumer Layer (a.k.a. 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 KPIs for each task or process could be defined in the QoS Layer. The aggregation of services could be guided by the Governance Layer.

All the services should be represented and described by the Service 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 provides 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 services 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, QoS Layer, as well as the Information Architecture 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 them 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 QoS 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 architectural building blocks of the Integration Layer. For example, a single or federated set of Enterprise Service Bus(es) 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 List of 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:

1. **Process Definition:** This category of capabilities is required for defining the business processes/operational flow of the business.
2. **Event Handling:** This category of capabilities handles business events in the context of a business process such as emitting/publishing event, subscribing/listening to business events.
3. **Process Runtime Enablement:** This category of capabilities enables BPM and helps realizing the business processes in runtime environment using standards such BPEL, SCA etc.
4. **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 repository of assets need by it.
5. **Decision Management:** This category of capabilities defines and manages the decision points and associated rules with in a business process.
6. **Process Integration:** This category of capabilities facilitates integration with others layer of SOA RA and helps exposing a business process as a service.
7. **Security and Policy Compliance:** This category of capabilities enables access control and policy enforcement in the business processes.
8. **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 following capabilities:

Process Definition

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

Event Handling

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

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 there in.
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 human.

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 & 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 bottleneck, and optimize work load assignments.

11.1.3 List of Architectural Building Blocks

The architectural building blocks 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
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**	QoS Layer: Policy Enforcer	17
19.		QoS Layer: Access Controller	18
20.	**Process Monitoring and Management**	QoS Layer: Business Activity Monitor	19

Table 4 ABB to Capability Mapping for the Business Process Layer

11.2 Business Process Layer: Description of ABBs and Supported Capabilities

11.2.1 ABBs Details

This section describes each of the ABBs in the integration 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 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 human 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 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 managing 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 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 a 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 endpoint that needs to serve the request based on the context. Context provides the extra information that this ABB needs to make the intelligent decision on which endpoint 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 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 Architecture 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 QoS Layer: Policy Enforcer

See QoS Layer Policy Enforcer ABB.

11.2.1.19 QoS Layer: Access Controller

See Access Controller ABB in the QoS Layer.

11.2.1.20 QoS Layer: Business Activity Monitor

See Business Activity Monitor ABB in the QoS 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:

1. Definition, composition and decomposition of a business process.
2. Handling of events.
3. Runtime enablement and realization of business processes.
4. Management of information and associated data flow in the context of the business processes.
5. Management of decision points/points of variability and associated rules in the context of the business processes.
6. Integration capabilities necessary for realizing business processes.
7. Security and policy compliance pertaining to business processes
8. Monitoring and management of business processes

The following figure illustrates the ABBs partitioned into key categories:

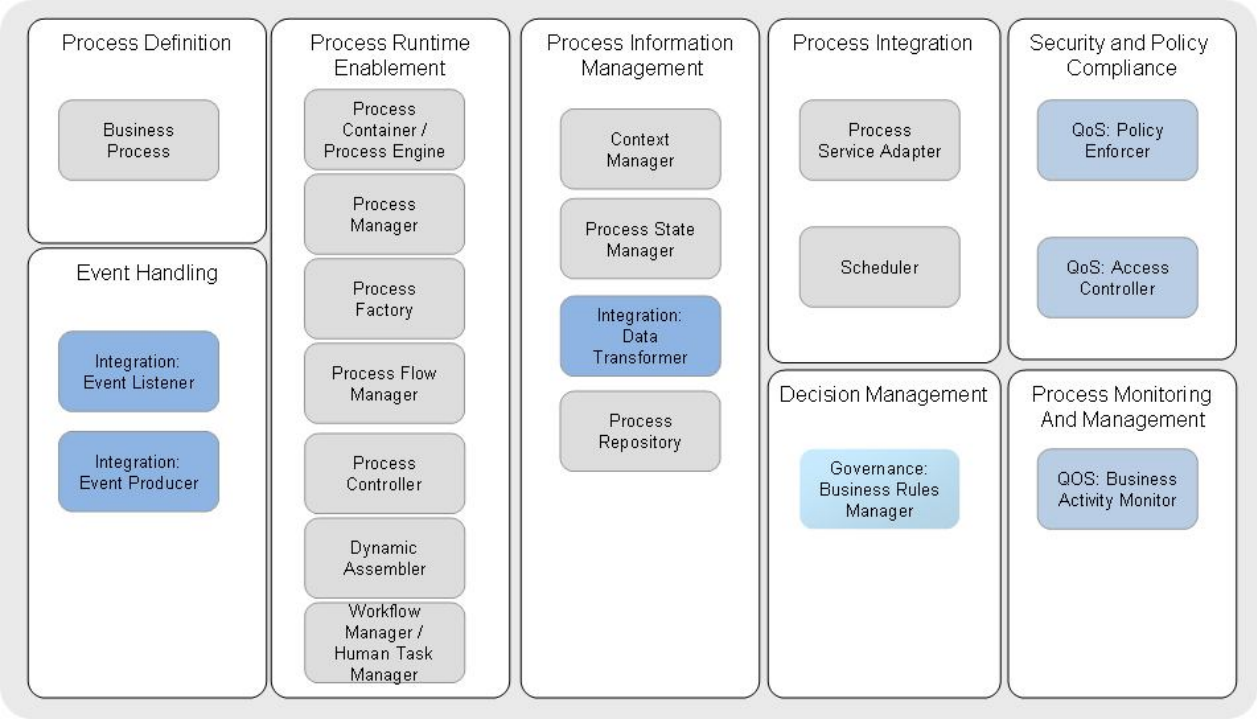


Figure 24 ABBs in the Business Process Layer

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 Business Process Layer: Interrelationships 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 RA layers support.

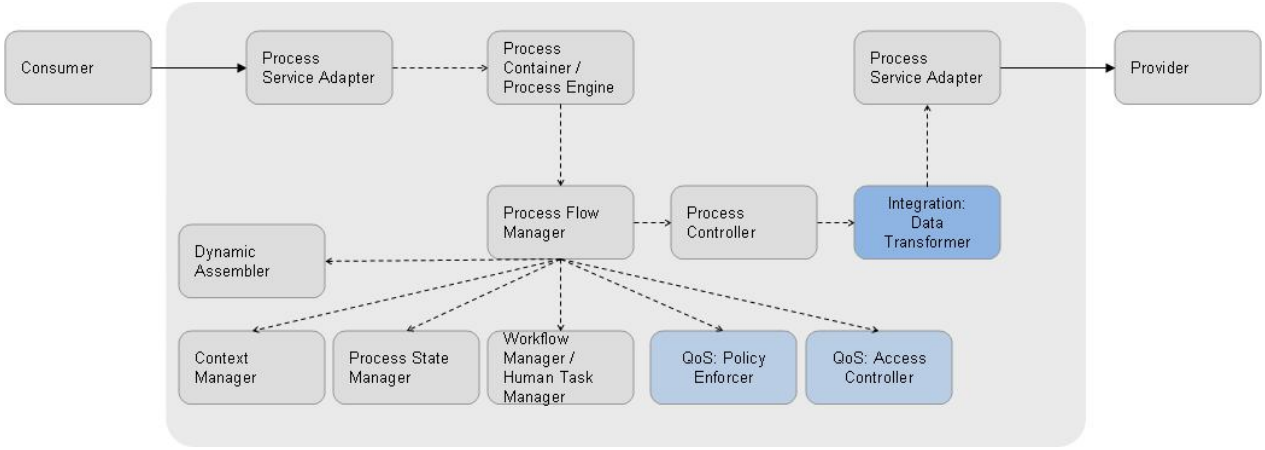


Figure 25 Key Relationships among ABBs in the Business Process Layer

11.4 Business Process Layer: 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 fulfil its responsibilities.

5. It relies on Governance Layer for following capabilities:
    a. Ability to store meta-data for policies.
    b. Ability to support management (storage, retrieval, etc.) of rules to support rules associated with decision points in service mediation, orchestration and composition.
6. It relies on QoS Layer for following capabilities:
    a. Ability to authenticate/authorize for service invocation.
7. It relies on Information Architecture Layer for following capabilities:
    a. Ability to store and retrieve meta-data and data required for the execution of the business processes.
8. It relies on Integration Layer for following capabilities:
    a. Ability to invoke services to realize systematic process steps.
    b. Ability to transform data from one format to another.

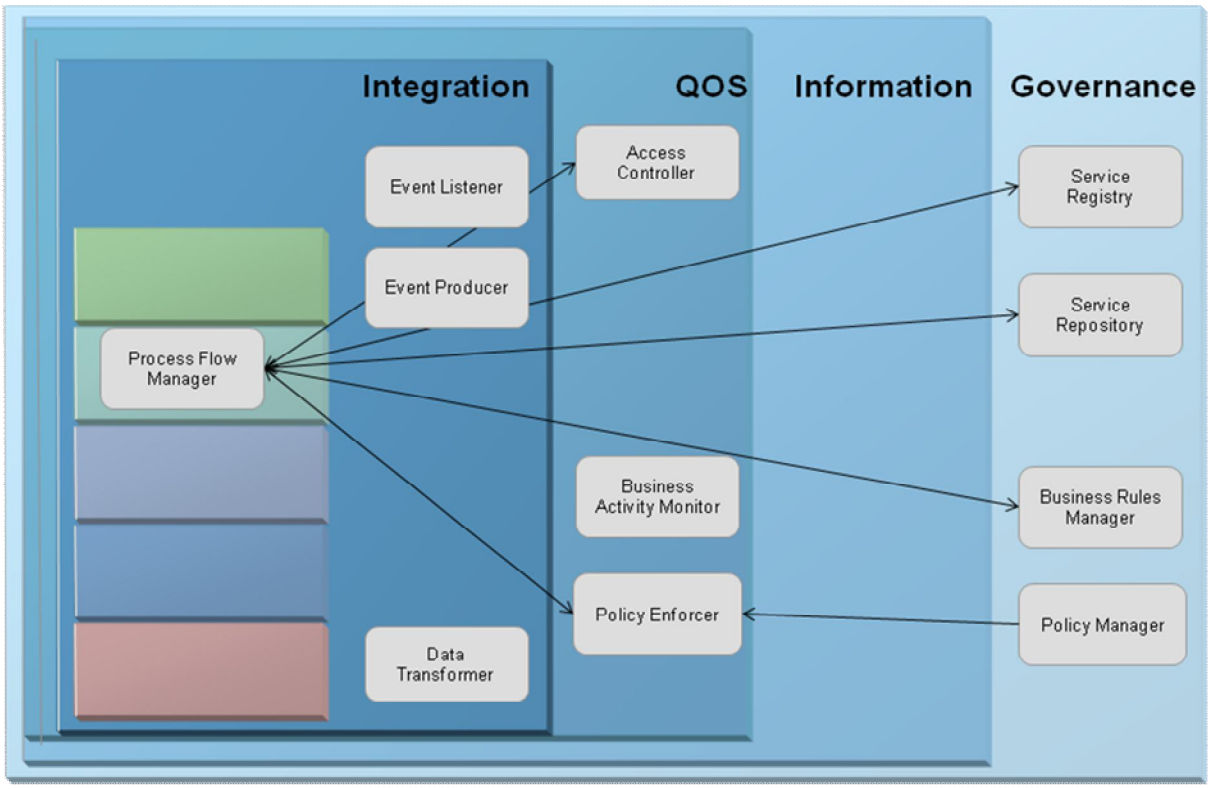


Figure 26 Key Interactions of the Business Process Layers with Cross Cutting Layers

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

e. It leverages Service Registry ABB and Service Repository ABB from the Governance Layer for storing meta-data such as policy, schema etc and for providing to access the meta-data. 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).

f. It leverages Business Rule Manager ABB in the Governance Layer to manage the rules supporting the decision points or points of variability with in a business process.

g. It leverages Access Controller ABB in the QoS Layer for authenticating/authorizing facility for service invocation and work load assignment.

h. It leverages Data Aggregator ABB, Data Federator AB, Data Consolidator ABB, Information Meta-data Manager ABB and Data Repository ABB from Information Architecture Layer to store and assess data.

i. It leverages Access Controller ABB in the QoS Layer to enforce access privileges and Policy Enforcer ABB in the QoS Layer to enforce policies.

j. It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Layer such as data transformation, service request etc. It leverages Mediator ABB in the Integration Layer to integrate with existing systems and

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

11.4.2 Interaction with Other Horizontal Layers

The Consumer Layer can invoke or trigger events that will execute business process 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 Service Layer.

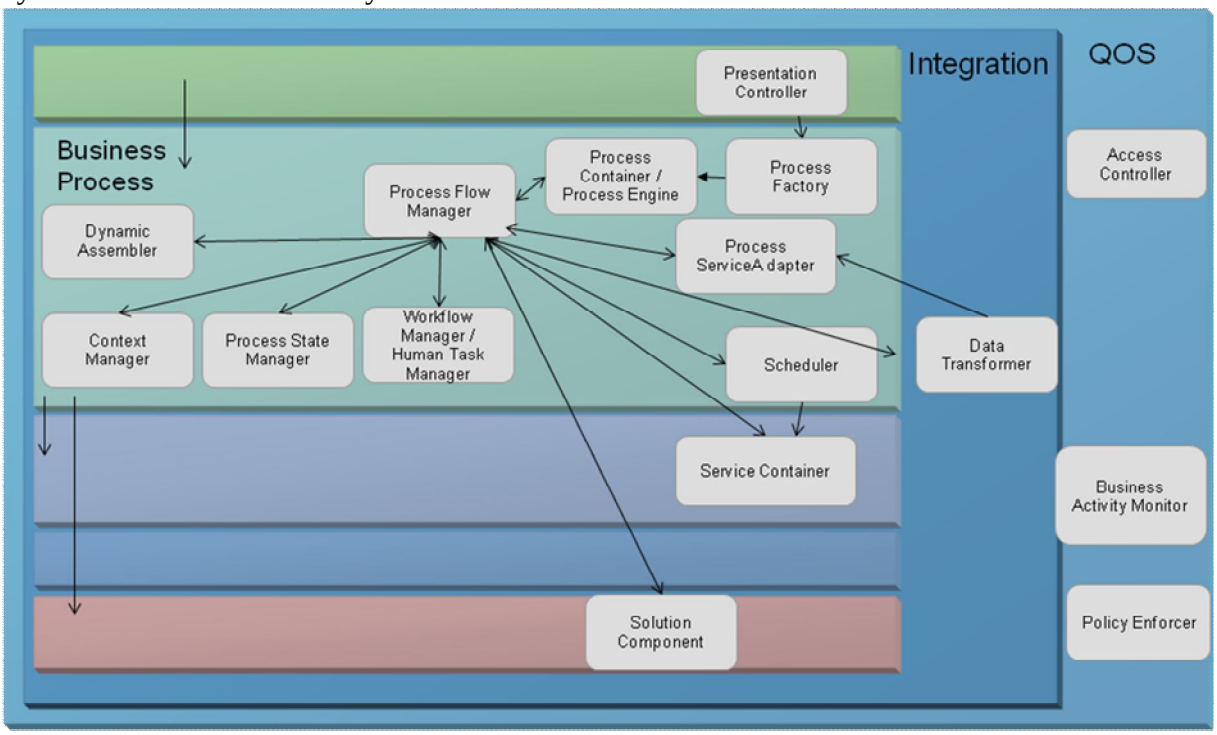


Figure 27 Key Interactions of the Business Process Layer with Other Horizontal Layers

11.5 Business Process Layer: 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 Layer 5: Consumer Layer

12.1 Consumer Layer: Overview

12.1.1 Context and Typical Flow

The Consumer Layer is the point where consumers interact with the SOA. It enables a 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. 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 SOA's, cloud service consumers, human users etc.

This decoupling between the consumer and the rest of the underlying SOA provides organizations the ability to support agility, enhanced reuse 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 [15], portlets, WSRP (Web Services for Remote Portlets 2.0)[16] and business to business integration interfaces. Thus the SOA enables us to cater to 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 [22]) 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 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 List of Capabilities

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

1. **Consumer Services:** This category of capabilities addresses the support of interaction with consumers.
2. **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.
3. **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.
4. **Caching and Streaming Content:** This category of capabilities addresses the support of information buffering and performance, and support the operation of the Consumer Layer.
5. **Security and Privacy:** Capabilities that address the support of quality of service, information protection and security.
6. **Information Access:** This category of capabilities addresses the sharing of data and meta-data across the layers of the SOA RA such as quality of service attributes, attributes defining common rules to be used across the layers etc.

This layer features 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 the ability to support customer-specific information, (enabled by the Information Architecture 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

15. Ability to access data and metadata through the Information Architecture Layer.

12.1.3 List of Architectural Building Blocks

The architectural building blocks 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
		Presentation Flow Manager	5
		Composite View	3
5.		Consumer/User Profile Manager	4, 6
6.		Personalization Manager	6
7.	**Backend Integration**	Integration Layer: Integration Controller	8
8.		Integration Layer: Data	9

		Transformer	
9.	**Caching and Streaming Content**	Cache	10 - 11
10.	**Security and Privacy**	Governance Layer: Policy Manager	12
11.		QoS Layer: Access Controller	12 - 14
12.	**Information Access**	Information Architecture Layer: Data Repository	15, 4

Table 5 ABB to Capability Mapping for the Consumer Layer

12.2 Consumer Layer: Details of ABBs and Supported Capabilities

12.2.1 ABBs Details

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 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 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.

12.2.1.4.1 Presentation Flow Manager

This ABB is responsible in 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.

12.2.1.4.2 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.

12.2.1.5 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.

12.2.1.6 Personalization Manager

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

1. 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
2. collaborative filtering technology based personalization to select content based on common interests or behaviours.

12.2.1.7 Integration Layer: Integration Controller

See Integration Controller ABB in the Integration Layer.

12.2.1.8 Integration Layer: Data Transformer

See Data Transformer ABB in the Integration Layer.

12.2.1.9 Cache

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

12.2.1.10 Governance Layer: Policy Manager

See Policy Manager ABB in the Governance Layer.

12.2.1.11 QoS Layer: Access Controller

See Access Controller ABB in the QoS Layer.

12.2.1.12 Information Architecture Layer: Data Repository

See Data Repository ABB in the Information Architecture 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:

1. Ability to support the interaction of the SOA with consumers.
2. Ability to support presentation, the creation of composite views and presentation control, content composition and decomposition, and the consumer centric configuration of views.
3. Ability to integrate information from services from the SOA and transform their information and incorporate it into content[5].
4. Ability to support information buffering and performance, and support the operation of the Consumer Layer.
5. Ability to support quality of service, information protection and security.
6. Ability to address the sharing of meta-data across the layers of the RA.

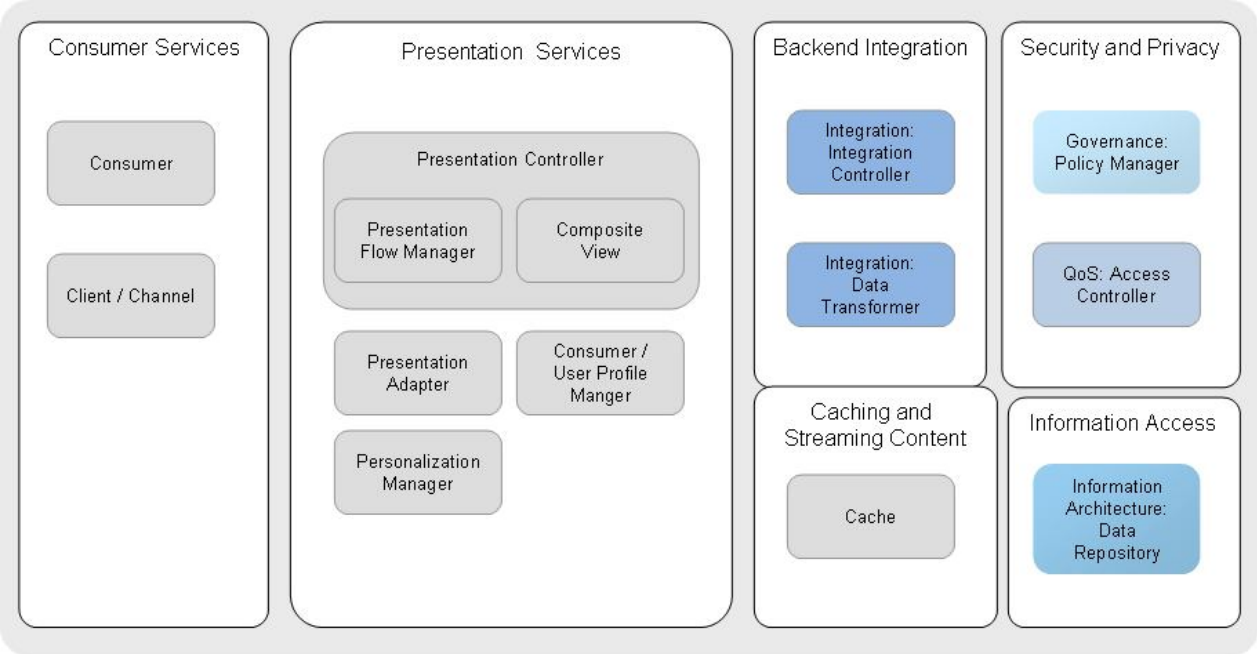


Figure 28 ABBs in the Consumer Layer

[5] 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

12.3 Consumer Layer: Interrelationships 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 quality of service), 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 reusability and agility, leveraging the core services in the Integration Layer and Business Process Layer and the capabilities of the rest of the SOA.

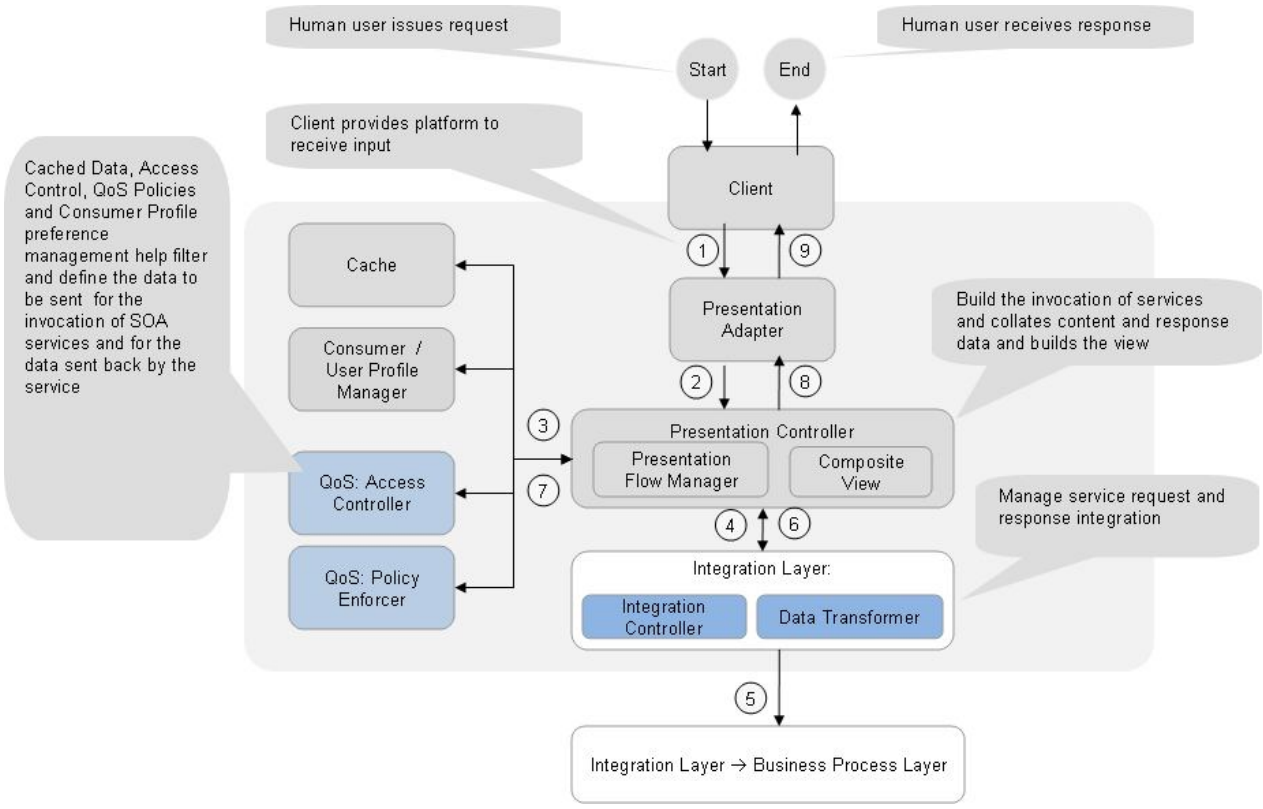


Figure 3 Interaction of a Human Service Consumer with the SOA via the Consumer Layer

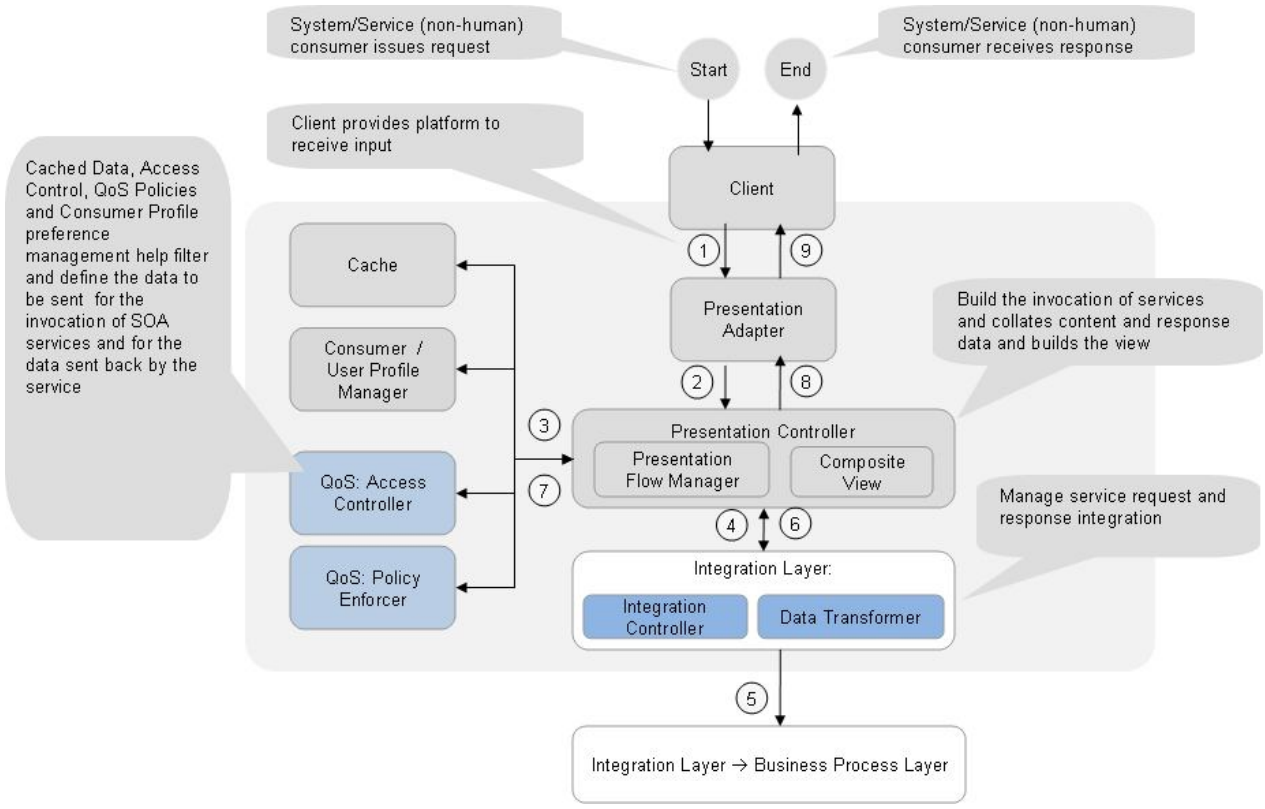


Figure 4 Interaction of a Systematic (Non-Human) Service Consumer with the SOA via the Consumer Layer

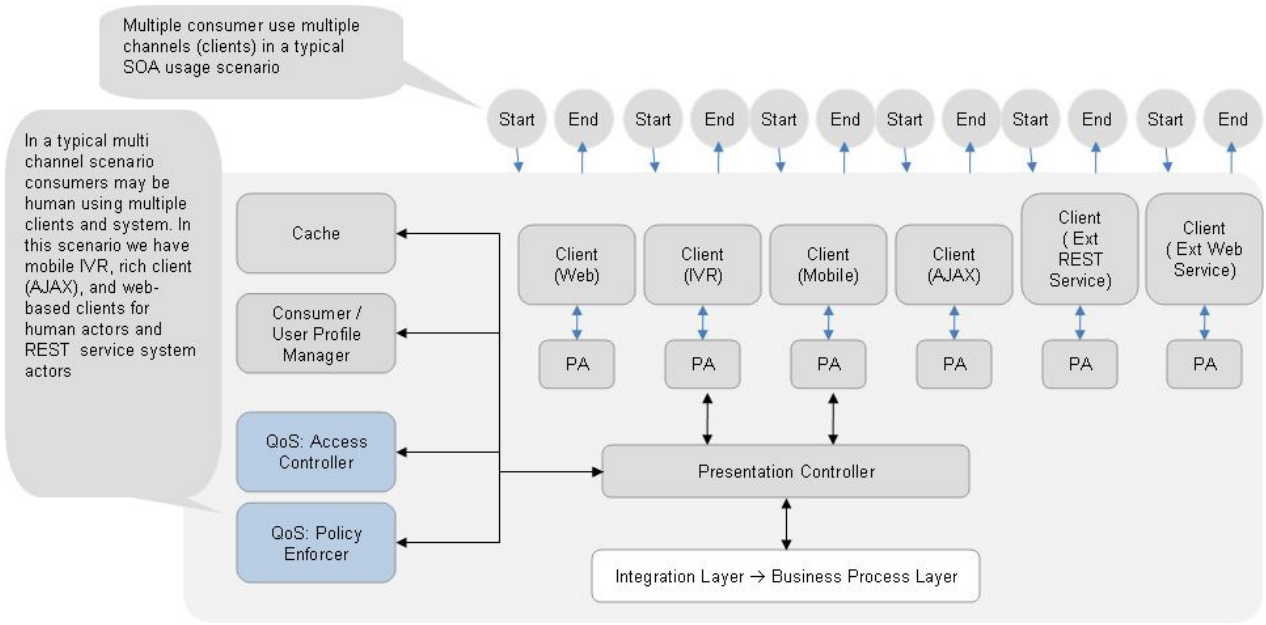


Figure 5 A Typical SOA Usage Scenario with Multiple Consumers using Multiple Channels

Figure 5 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 Consumer Layer: 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 fulfil its responsibilities.

4. It relies on Governance Layer for following capabilities:
    a. Ability to store meta-data for policies.
5. It relies on QoS Layer for following capabilities:
    a. Ability to authenticate/authorize for service invocation.
6. It relies on Information Architecture Layer for following capabilities:
    a. Ability to store and retrieve meta-data and data.
7. It relies on Integration Layer for following capabilities:
    a. Ability to invoke business processes and/or services.
    b. Ability to transform data from one format to another.

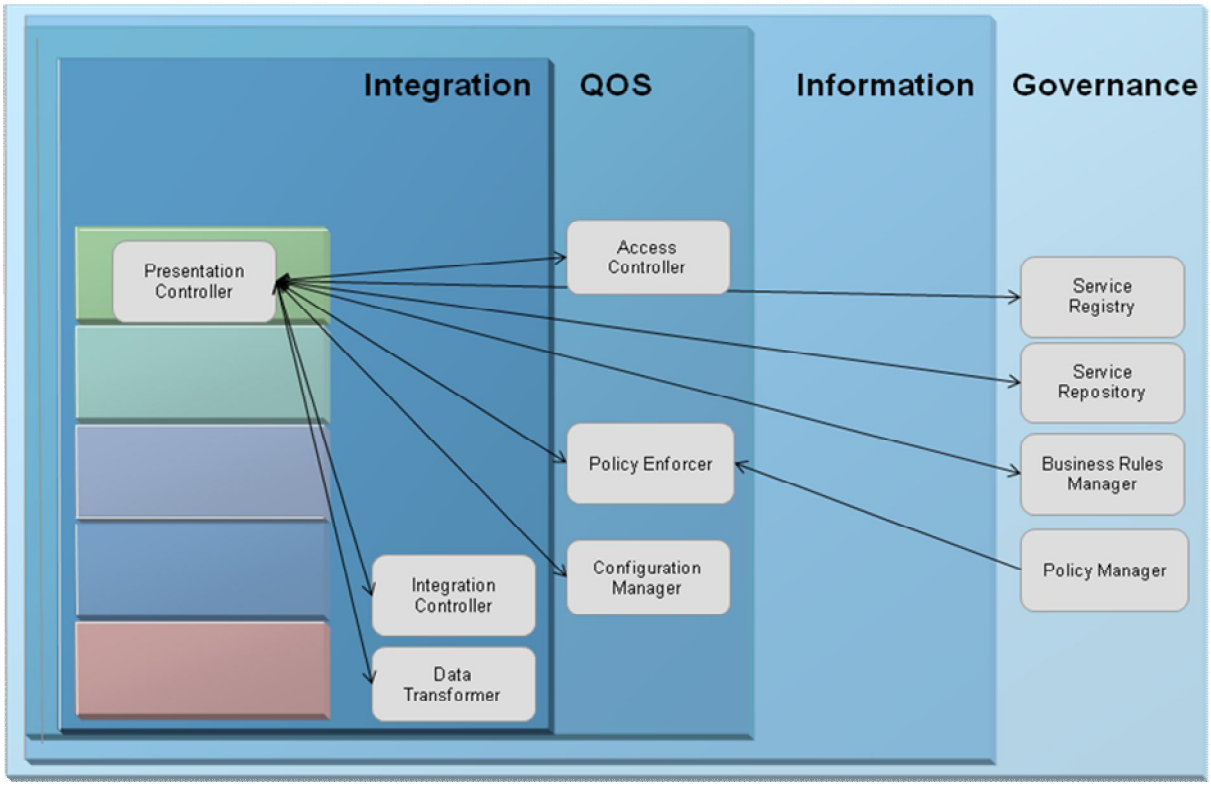


Figure 29 Key Interactions of the Consumer Layer with the Cross Cutting Layers

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

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

12.4.2 Interaction with Other Horizontal Layers

Within the Consumer Layer, it primarily interacts with the Consumer Profile Manager ABB and the Presentation Control ABB and contains configuration information for the other ABBs in the Consumer Layer.

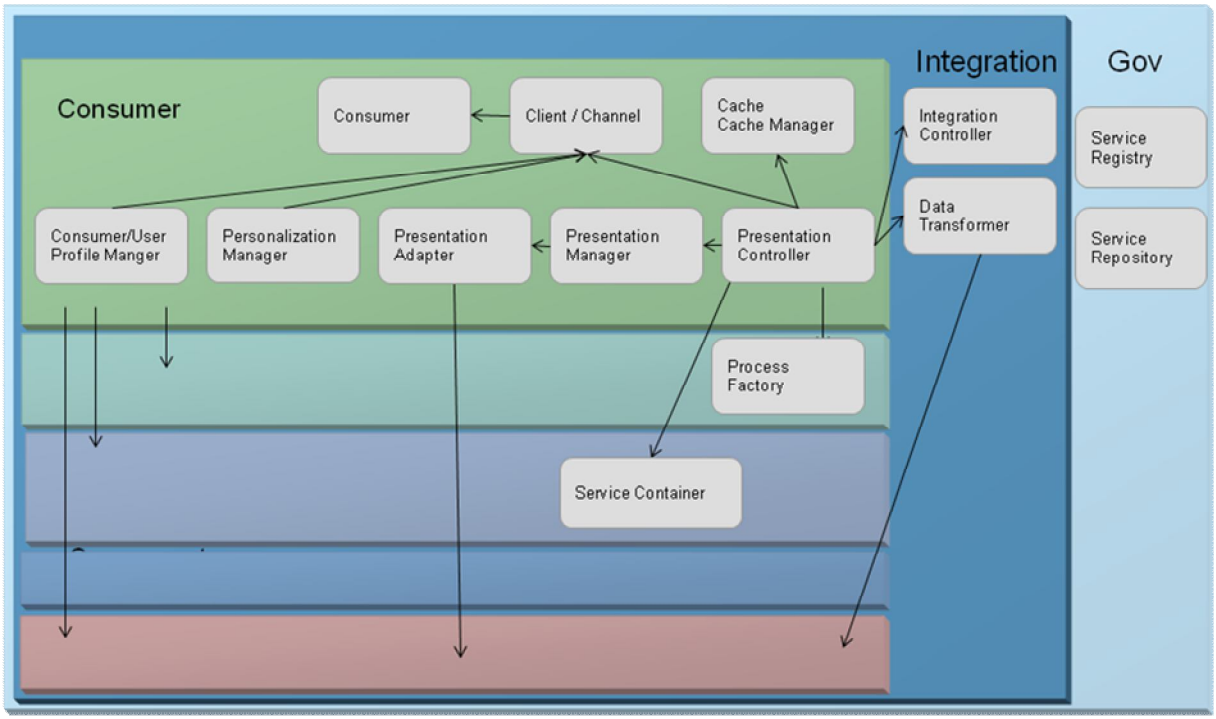


Figure 30 Key Interactions of the Consumer Layer with the Other Horizontal Layers

12.5 Consumer Layer: Usage Implications and Guidance

This layer provides the SOA 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 to be used. The architecture lets organizations and industry organizations maintain consistent standards and common implementations.

13 Layer 6: Integration Layer

13.1 Integration Layer: Overview

13.1.1 Context and Typical Flow

The Integration Layer is a key enabler for a 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 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 endpoints 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 thru 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), Voice XML (for IVR), XML String, can be done via XSLT functionality supported through message transformation capability in the integration layer.

As shown in Figure 1, 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 themselves.
- Consumers and providers are decoupled; this decoupling allows integration of disparate systems into new solutions.

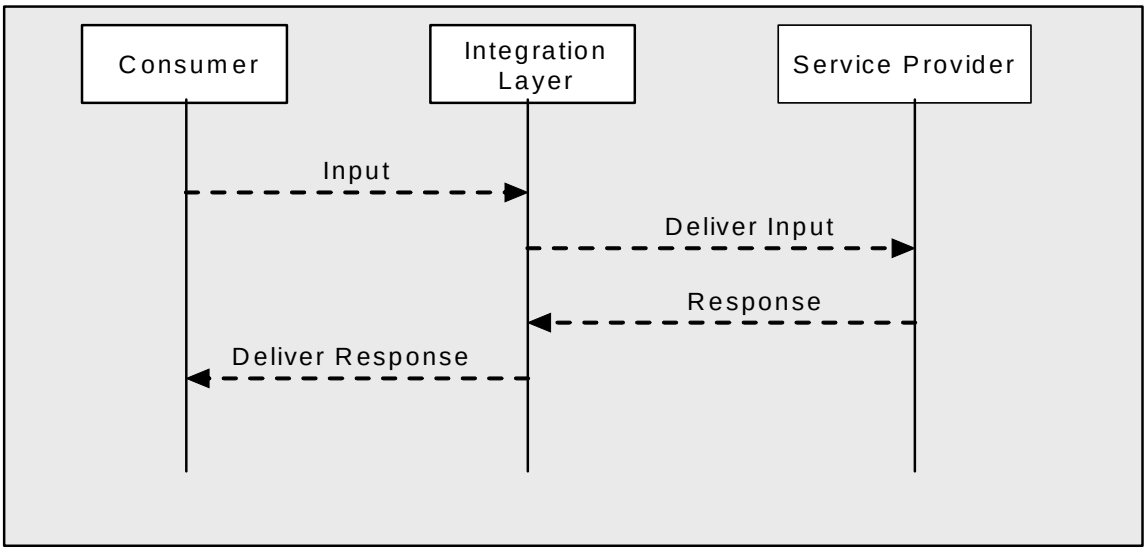


Figure 31: Usage of the Integration Layer

13.1.2 List of Capabilities

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

1. **Communication, Service Interaction, and Integration**: This category of capabilities provides ability to route request to correct provider after necessary message transformation and protocol conversion and to connect service requestor to the service provider and its underlying solutions platforms realizing the requested service. It also provides 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.
2. **Message Processing**: This category of capabilities provides ability to perform the necessary message transformation to connect service requestor to the service provider and to publish and subscribe messages and event asynchronously.

3. **Quality of Service**: This category of capabilities supports handling of transaction and exceptions and other non functional requirements.
4. **Security**: This category of capabilities helps in enforcement of access privileges and other security policies.
5. **Management**: This category of capabilities provides ability to maintain service invocation history and monitor and track the service invocations.

This layer features 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 variety of protocols.
4. Ability to support variety of messaging styles such one way, pub-sub, request-response.
5. Ability to route messages to 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 List of Architectural Building Blocks

The architectural building blocks 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**	QoS Layer: Access Controller	15
17.	**Management**	Logger	16
18.		Auditor	17
19.		QoS Layer: Configuration Manager	18

Table 6 ABB to Capability Mapping for the Integration Layer

13.2 Integration Layer: Description of ABBs and Supported Capabilities

13.2.1 ABBs Details

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

13.2.1.2 Adapter

This ABB is responsible for 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 to handling the service request/response interaction. It also supports the transformation between message formats, conversion of protocol, 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.

13.2.1.3.1 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 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 Message Transformer ABB, Auditor ABB, Logger ABB, and Exception Handler ABB. It may leverage Access Controller ABB from QoS layer.

13.2.1.3.2 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.

13.2.1.3.2.1 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.

13.2.1.3.2.2 Data Transformer

This ABB is responsible to transform 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 meta-data such as the canonical form etc.

13.2.1.3.3 Protocol Converter

This ABB is responsible to transforms 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.

13.2.1.4 Asynchronous Messaging Manager

This ABB is responsible to store and forward message potentially using a message queue and it provides the underlying plumbing to transport messages between service invokers 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.

13.2.1.5 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.

13.2.1.6 Data Aggregator

This ABB supports the ability to compose (aggregate) data from different services/service providers which are interacting with integration layer using 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.

13.2.1.7 QoS Layer: Access Controller

See Access Controller ABB in the QoS Layer.

13.2.1.8 Logger

This ABB is responsible to capture 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 QoS layer's monitoring features. It is important from a monitoring and support for compliance perspective.

13.2.1.9 Auditor

This ABB is responsible to track and monitor 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.

13.2.1.10 Exception Handler

This ABB is responsible to handle 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.

13.2.1.11 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 Asynchronous Messaging Manager ABB, Messaging Transformer ABB, and Router ABB.

13.2.1.12 Event Producer

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

13.2.1.13 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.

13.2.1.14 QoS Layer: Configuration Manager

See Configuration Manager ABB in the QoS 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:

1. 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.

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

   b. Ability to mediate reliability through the imposition of WS* and other standards based protocol.

   c. 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.

2. Ability to perform message transformation.

3. Ability to support quality of service requirements such as transaction management, performance criteria, exception handling etc

4. Ability to support security requirements.

5. Ability to track, monitor and mange service invocations.

The following figure illustrates the ABBs partitioned into key categories:

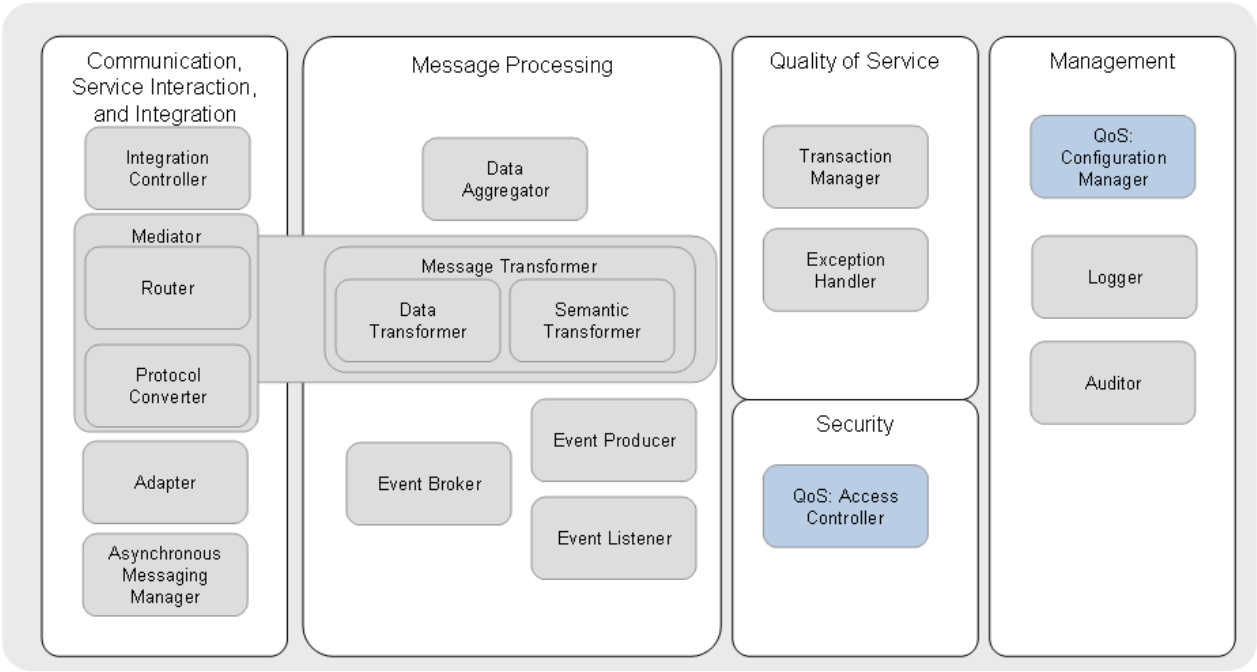


Figure 32 ABBs in the Integration Layer

13.3 Integration Layer: Interrelationships 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 RA layers support.

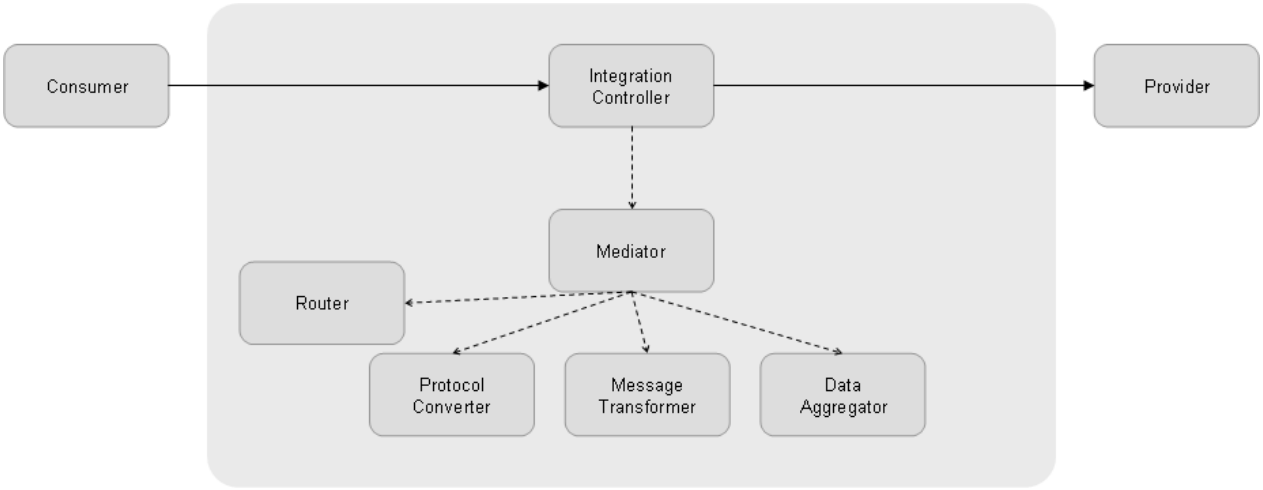


Figure 33 Simple Interactions between Consumer and Provider through the Integration Layer

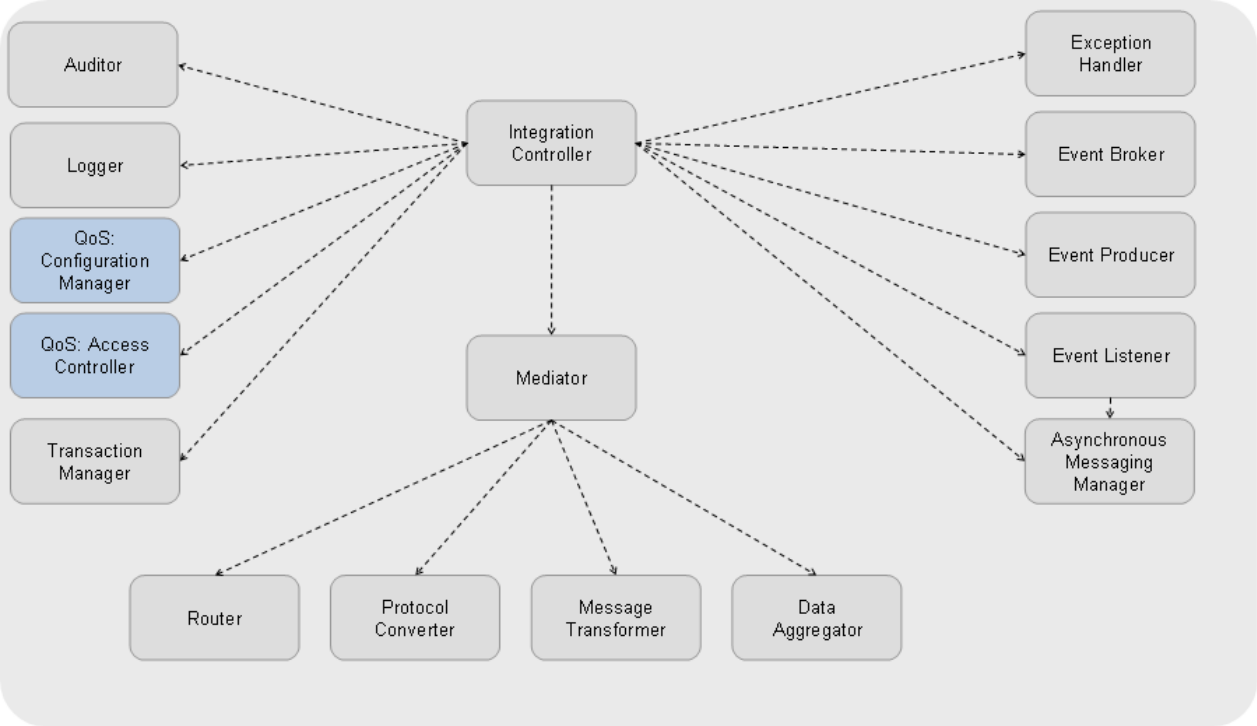


Figure 34 Relationships among ABBs in the Integration Layer

13.4 Integration Layer: Significant Intersection Points with other Layers

13.4.1 Interaction with Other Cross Cutting Layers

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

1. It relies on Governance Layer for following capabilities:
    a. Ability to store meta-data for policies.
    b. 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).
    c. Ability to determine service end-points for service virtualization.
2. It relies on QoS Layer for following capabilities.
    a. Ability to authenticate/authorize for service invocation and messages.

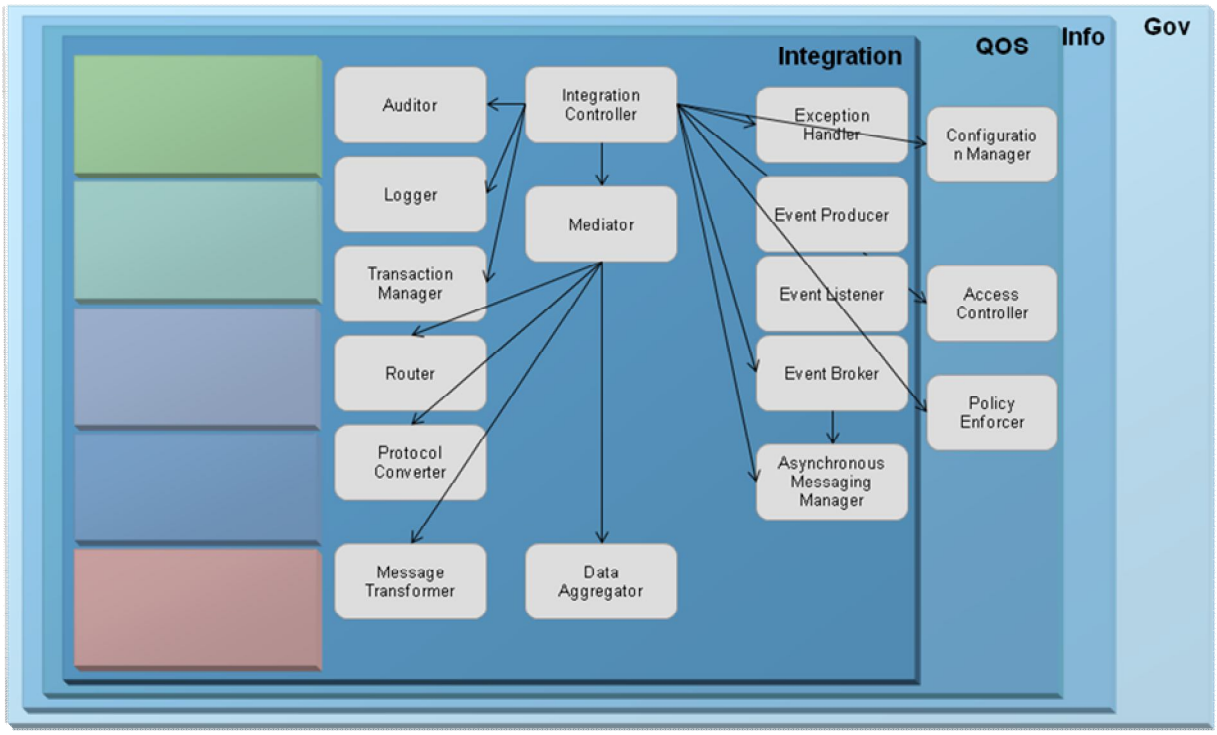


Figure 35 Key Interactions of the Integration Layer with the Other Cross Cutting Layers

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

a. It leverages Service Registry ABB and Service Repository ABB from the Governance Layer for storing meta-data such as policy, schema etc and for providing to access the meta-data. Service Registry ABB contains service definitions at runtime and supports service virtualization and service discovery.
b. It leverages Business Rule Manager ABB in the Governance Layer to support rule implementation for the Integration Layer.
c. It leverages Access Controller ABB in the QoS Layer for authenticating/authorizing facility for service invocation and message routing. It also leverages Policy Enforcer ABB in the QoS Layer to enforce policies local to the Integration Layer.
d. Data Transformer ABB in the Integration Layer uses meta-data from Information Architecture Layer and leverages Information Meta-data Manager ABB from the Information Architecture Layer for data transformation.
e. Event Broker ABB, Event Listener ABB, Event Producer ABB in the Integration Layer uses Event Manager ABB in the Governance Layer for event definition and related information.

13.4.2 Interaction with Horizontal Layers

Horizontal layers of SOA RA leverage ABBs from this layer to provide their respective capabilities. Horizontal layers of the SOA RA use 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.

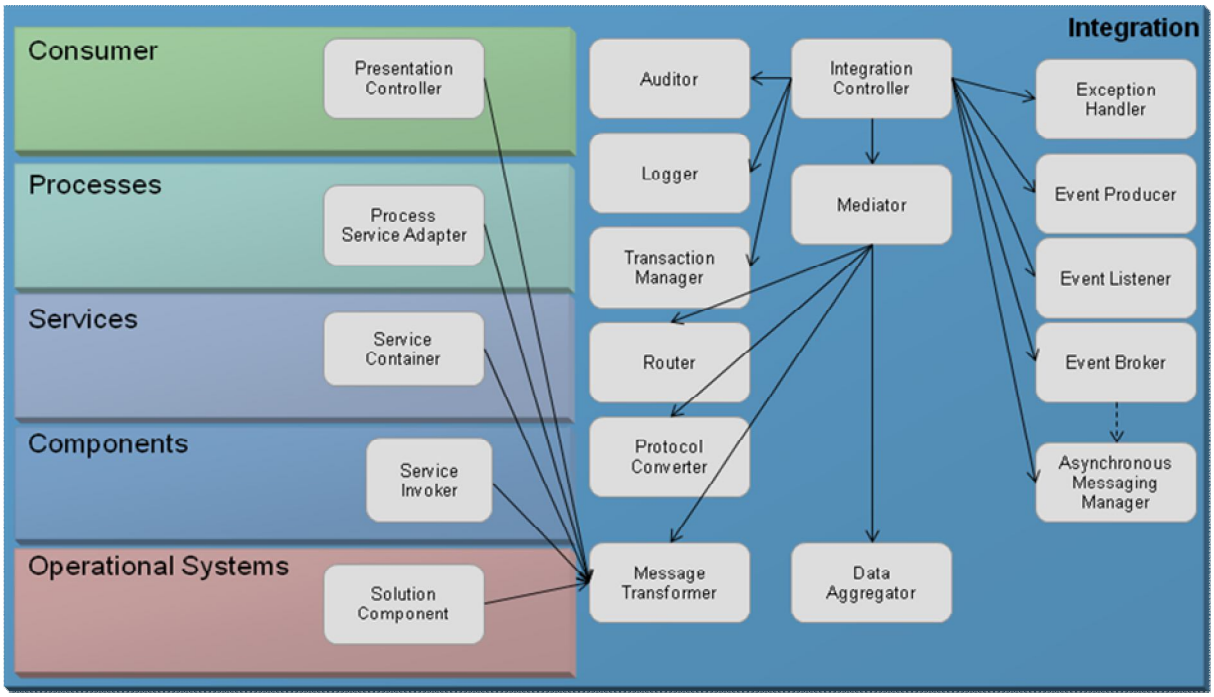


Figure 36 Key Interactions of the Integration Layer with the Horizontal Layers

The following figure provides a typical interaction with horizontal layers of SOA RA with the Integration Layer:

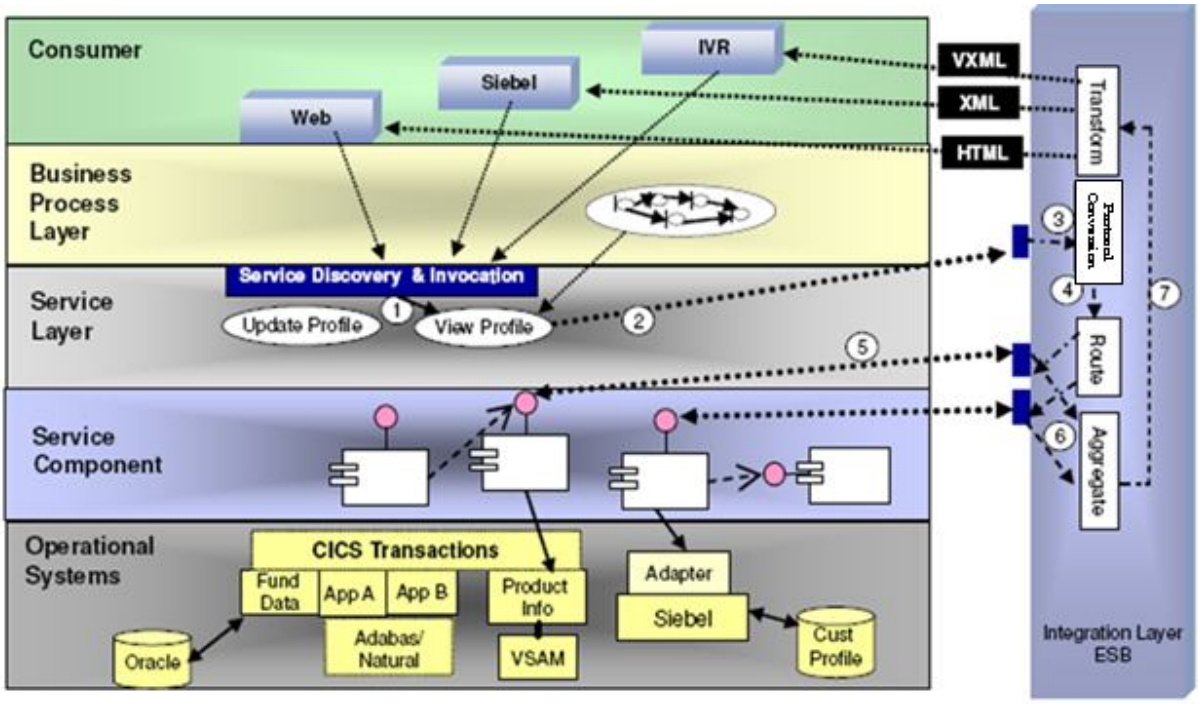


Figure 37 Detail Interactions of the Horizontal Layers with the Integration Layer

In this example above,

1. Through the service discovery and invocation process, the web client looks up the View Profile Service.
2. A connection is made to the Integration Layer (ESB).
3. The ESB performs protocol conversion, if necessary
4. It subsequently routes the call to the appropriate destination.
5. It then receives the results of the call
6. It then aggregates the results of the call.
7. 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 Integration Layer: 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 Layer 7: Quality of Service (QoS) Layer

14.1 Quality of Service Layer: 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, heterogeneous computing infrastructures, decentralized 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 QoS Layer include:

- monitoring and management both at the 1) business level in terms of KPIs, events, and business activities in the business process 2) IT systems level for the security, health and well being of IT systems, services, applications, networks, storage, and compute servers.
- enforcement of multitude of policies 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, NFRs, and governance regimens are adhered to.

This layer supports monitoring and capturing service and solution metrics in an operational sense and signalling non-compliance with non-functional requirements (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 signalling a non-compliance condition with salient policies is detected or (preferably) when a non-compliance condition is anticipated.

This layer supports the following capabilities to enable quality of service management for services and solutions:

1. Ability to monitor and capture metrics and status
2. Ability to find and access policies
3. Ability to evaluate and enforce policies at check points or on metrics captured
4. Ability to send notification and log of non-compliance
5. Ability to change the rules, policies, configuration and status
6. Ability to support solution, business activity and IT management
7. Ability to secure the solution

The Governance Layer defines the non functional requirements and policies for the SOA solution which are captured and enforced in the QoS layer. Responses (dispensations and appeals) to non compliance exceptions are defined by the Governance Layer as well.

Important areas of policy enforcement are security, messaging transportation, and infrastructure availability, and service availability. These supporting security management and systems management for SOA solutions.

14.1.2 List of Capabilities

There are a set of categories of capabilities that Quality of Service layer needs to support in the SOA RA. These categories are:

- Abilities to monitor and manage IT Systems
- Abilities to monitor and manage solution status
- Abilities to monitor and manage business activities
- Abilities to change solutions configuration and descriptions
- Abilities to monitor and enforce policy and business rules
- Abilities to monitor and manage security
- Abilities to signal and record compliance status or metrics

This layer features 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, suchas 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 servicesare 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 thatcontribute 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 Management

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

Application Systems Management and Monitoring

26. Ability to coordinate overall QoS requirements for the SOA solution.
27. Ability to desribe QoS non functional requirements.

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 availabilty metrics.
35. Ability to evaluate the availabiltiy metrics against non functional requirements (policy).
36. Ability to capture metrics on performance of services and solutions
37. Ability to change configuration and policy to ensure meeting service level agreements
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 specifc 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.

Update Management

46. Ability to capture configuration (authoring tools).
47. Ability to change configuration.
48. Ability to check QoS requirements for valid configurations.
49. Ability to change configuration to comply with QoS requirements.
50. Ability to send events for non compliance to QoS requirements.
51. Ability to track and record changes, configuration, metadata, policy, etc., happening in the solution.
52. Ability to reverse changes in the solution.
53. Ability to ensure that changes are executed in compliance with relevant governance policies.
54. Ability to change metadata, including service descriptions
55. Ability to propagate metadata changes to other repositories and descriptions.

Event Management

56. Ability to control issuance of events in the solution.
57. Ability to send issue events indicating non compliance to QoS requirements.
58. Ability to subscribe to events issued by the solution.
59. Ability to log events and business messages.
60. Ability to control logging frequency and size.

Policy Monitoring and Enforcement

61. Ability to check QoS requirements for valid rules.
62. Ability to change rules to comply with QoS requirements.
63. Ability to change QoS requirements to comply with rules.
64. Ability to send events for non compliance to QoS requirements.

65. Ability to evaluate policies.
66. Ability to resolve conflicts between policies.
67. Ability to enforce compliance with policies
68. Ability to automatically respond and correct violations of policies (enforce).
69. Ability to enable policy enforcement
70. Ability to discover, analyze, transform, distribute, evaluate and enforce security policies.
71. Ability to manage non functional QoS solution requirements from solution delivery to solution termination.
72. Ability to manage life cycle of policies.
73. Ability to represent policies.
74. Ability to author policies.
75. Ability to manage instances of policies.
76. Ability to change policies.
77. Ability to disable, discard, discontinue policies.
78. Ability to monitor and capture metrics and status
79. Ability to find and access policies
80. Ability to evaluate policies at check points or on metrics captured
81. Ability to automate monitoring for violations of policy.
82. Ability to author, discover, analyze, transform, distribute, evaluate and enforce security policies

Data Storage

83. Ability to store QoS policies and rules
84. Ability to locate/find/return QoS policies and rules.

Capabilities from the SOA RA integration layer needed to enable QoS Layers

- Service Integration (adapters)
- Service Mediation
- Message Routing and Transport
- Asynchronous Messaging
- Transaction Management
- Data Aggregation
- Message, Semantic, Data and Protocol Transformation
- Exception Handling

14.1.3 List of Architectural Building Blocks

The architectural building blocks responsible for providing these sets of capabilities in the information 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.		Security Policy Manager	12
		Identity, Access and Entitlement Management	13
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, All
12.		Data Driven Access Controller	22
13.	**IT Systems Management**	IT Systems Manager	23-25
14.		Systems Manager	23
15.		Network Manager	24
16.		Storage Manager	25
17.	**Application Systems Management**	Solution Manager	26 - 27
18.		Status Manager	41 - 44
19.		Monitoring Metric Tools	45
20.		Lifecycle Manager	28
21.		Performance Manager	36 - 39
22.		Reliability Manager	29 - 30
23.		Availability Manager	31 - 35
24.		Execution Cost Manager	40
25.	**Business Activity Management and Monitoring**	Business Activity Manager	
26.		Business Activity Monitor	
27.		Activity Correlation Manager	
28.	**Update Management**	Configuration Manager	46 - 50

29.		Metadata Manager	44 - 45
30.		Governance: Change Control Management	51 - 53
31.	**Event Management**	Logging Manager	59 - 60
32.		Event Manager	56 - 58
33.		Integration Layer: Event Listener	58
34.		Integaction Layer: Event Producer	56
35.	**Policy Monitoring and Enforcement**	Policy Enforcer	65 - 69; All
36.		Security Policy Enforcer	70
37.		Policy Monitor	78 - 81
38.		Governance Layer: Policy Manager	71 - 77
39.		Governance Layer: Business Rules Manager	61 - 64
40.		Governance Layer: Security Policy Manager	82
41.	**Data Storage**	Governance: Repository	83 - 84

Table 7 ABB to Capability Mapping for the QoS Layer

14.2 Quality of Service Layer: Details of ABBs and Supported Capabilities

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

ABBs Details

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. It supports the ability to:

1. approve authority for security.
2. ensure that physical and operational security is maintained for locations, assets, humans, environment and utilities.
3. provide surveillance and monitoring of locations, perimeters and areas.
4. enforce entry control.
5. provide for positioning, tracking and identification of humans and assets; continuity and recovery operations.

14.2.1.2 Safety Manager

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

14.2.1.3 Security Aware Physical Asset Manager

This ABB supports the ability to secure physical assets, suchas locations, facilities, services, inventory, physical access control, human identity, etc.

14.2.1.4 Security Manager

This ABB is responsible for handling the security feature of a solution. Since a solution here refers to an SOA-oriented business solution, a solution is considered with high security if 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. The Security ABB is contained within the Service Container and is the interface with the Quality of Service (QoS) Layer's Security capabilities. It provides the binding to whatever standards the policies are written in the QoS layer ABB's and the ability to enforce them (acting as a Policy Enforcer for security policies).

14.2.1.5 Security Policy Manager

This ABB supports the management of policy needed to support security . A key tenant of an effective security program is management of security through a policy. This includes:

1. The ability to author policy
2. Policy discovery when new assets are introduced to the system
3. Policy analysis to evaluate policy changes or compliance
4. Policy transformation when adopting a new or augmented policy
5. Policy distributution to ensure the policy is deploed as appropriate
6. Policy evaluatation to determine action to take based on the policy
7. Policy enforcement to ensure adherance to security policies.

14.2.1.6 Identity, Access and Entitlement Manager

Identity, access and entitlement management provides trust management, identity lifecycle management, credential management, role entitlement and compliance management. This is concerned with the ability to assure that access to resources has been given :

1. To the right identities
2. At the right time
3. For the right purpose.

It also provides the ability to monitor and audit access to resources for unauthorized or unacceptable use

14.2.1.7 Data and Information Protector

This ABB protects unstructured and structured data from unauthorized access and data loss while providing the ability to monitor and audit access to information.

14.2.1.8 Software System and Service Assurer

The ABB supports the 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. This includes:

1. Structured design process
2. Threat modeling
3. Risk assessment
4. Design reviews for security
5. Source code review
6. Source code analysis
7. Dynamic application analysis
8. Source code control
9. Access monitoring;
10. Code/package signing and verification
11. Quality assurance testing;
12. Supplier and third party code validation
13. Security problem & incident management of software and services

14.2.1.9 Threat and Vulnerability Manager

This ABB supports the 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. This includes Vulnerability testing, vulnerability scanning, virtual patching and risk analysis

14.2.1.10 Risk and Compliance Assessor

The ABB supports the ability to identify, quantify, assess, and report on IT related risks thatcontribute to the enterprise's operational risk by providing all services to analyze and report security information and security events, and create alarms and insight. This includes:

1. Security and compliance dashboard
2. Forensics
3. Reporting
4. Compliance audit
5. Analytics

14.2.1.11 Security Aware Service Manager

This ABB supports the IT Service Management automation basis for security management, including tie backs to service management disciplines such as:

1. Incident management
2. Problem management
3. Change management
4. Release management
5. Asset management

14.2.1.12 Access Controller

This ABB acts is a kind of policy enforcer providing access control and enforcing policies related to access control. 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 the federated authentication (SSO) and the ability to ensure that the appropriate audit logging is carried out.

This ABB supports integration with the governance layer, which defines security policies, to retrieve security policies and act as a local Policy decision point and local Policy enforcement point.

It can include support for standards such as SAML (authentication and authorization)[], XDAS [] and CBE [] (audit and logging).

14.2.1.13 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 claim processing scenario by an insurance provider, tax identification number of a claimant is only to be viewed by set of individuals who are certified to handle sensitive personal information.

14.2.1.14 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 the Operational Systems Layer.

14.2.1.15 Systems Manager

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

14.2.1.16 Network Manager

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

14.2.1.17 Application Systems 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 non functional requirements (performance), must handle the information correctly (integrity), and must be able to recover that data that it has (reliability). Integrity and reliability are typically handle inside the application which uses several redundant storage and commit mechanism to achieve integrity and reliability. On the other hand, the availability and performance of the application depends on its components that supports the application and relationship and interconnection among the components.This ABB is responsible to understand these relationships and present 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.

14.2.1.18 Storage Manager

Management of storage resources, including access to local, networked, and virtualized storage.

14.2.1.19 Solution Manager

This ABB is the center of the QoS layer. It coordinates the managemen of the solution and all of the other ABBs. The solution manager is responsible for coordinating solution lifecycle, security, availability, configuration and change.

14.2.1.20 Status Manager

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

14.2.1.21 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.

14.2.1.22 Lifecycle Manager

This ABB is responsible for manage 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.

14.2.1.23 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. The performance manager can change configuration and policy to ensure meeting service level agreements 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 Reliability Manager

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

14.2.1.25 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.
.

14.2.1.26 Execution Cost Manager

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

14.2.1.27 Configuration Manager

This ABB is a set of tools used to define the configuration of SOA solution and processes being governed, as well as 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. (a) 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 be created and stored in a repository to be selected when needed. (b) Through dynamic template creation, where a user selects certain characteristics and the system will determine the appropriate rules and configure using relevant ABBs at run time. For instance, user may require that the system adopt industry messaging standard and satisfy some service level agreements (SLAs). Based on these requirements the system during run-time will select the appropriate data transformation, protocol conversion, and service providers that meet the SLAs.

14.2.1.28 Governance: Change Control Manager

See Governance Layer Change Control Manager ABB.

14.2.1.29 Meta Data Manager

This ABB is responsible for managing metadata in repositories

14.2.1.30 Logging Manager

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

14.2.1.31 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.

14.2.1.32 Integration Layer: Event Listener

See Event Listener ABB in the Integration Layer.

14.2.1.33 Integration Layer: Event Producer

See Event Producer ABB in the Integration Layer.

14.2.1.34 Governance Layer: Business Rules Manager

See Business Rules Manager in the Governance Layer.

14.2.1.35 Policy Enforcer

Used by Operational Systems, Service, Service Component, Business Process, Integration, Governance Layers

This ABB is responsible for enforcing quality of service policy and policy enforcement points in all the other functional and cross cutting layers. 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 SOA RA represent the Policy Enforcement Points in the architecture.

For example, in the Service layer, this ABB resides in the service container and provides the ability to enforce policies, ensuring the services comply with their service level agreements (SLAs). It supports the monitoring and management aspects of the services layer.

The Policy Manager in the governance layer sets and updates the policies to be enforced. The Policy Monitor ABBs in the Governance layer monitor that the policy enforcers are executing corrently.

14.2.1.36 Security Policy Enforcer

This ABB supports the ability to enforce security policies and is a type of policy enforcer.

14.2.1.37 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. A policy monitor obtains its policies from the Policy manager. The policy monitor is passive and interacts with the policy enforcer ABB to take any actions when violations are detected. It deals essentially with the monitoring and management aspects of the Services Layer.

14.2.1.38 Governance Layer: Policy Manager

See Policy Manager ABB in the Governance Layer.

14.2.1.39 Governance Layer: Security Policy Manager

See Security Policy Manager ABB in the Governance Layer.

14.2.1.40 Governance Layer: Business Rules Manager

See Business Rules Manager ABB in the Governance Layer.

14.2.1.41 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.

14.2.1.42 Business Activity Manager

This ABB enables the event information to be analysed, both in realtime/near-realtime, as well as stored (warehoused) events using the Activity Correlation ABB. It provides event-based analytic functionality, the ability to perform scenario analysis, and sense and respond capability. It uses the Activity Correlation Manager 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 pro-actively 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.

14.2.1.43 Activity Correlation Manager

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

14.2.2 Structural Overview of the Layer

The Quality of Service layer's components can be thought of as being logically partitioned into components which support

1. The management of Security Policies and Identity within the context of the SOA RA and its layers. It thus acts as a Policy Definition as well a Policy Enforcer, and must support the specification of various policies. Many of these policies may however be enforced by the associated layer.

2. The management of Quality of Service Policies for the runtime SOA RA. This would include policies dealing with availability, compliance, safety, etc for SOA solution processes as well as the IT systems.

The figure below illustrates the different ABBs and their interdependencies.

Figure illustrating the ABBs, partitioning into 3 sets:

1. Those that provide the key functionality within the layer
2. Those that provide the interface to other layers
3. Those that "manage" or orchestrate other ABB's, to provide a practical dynamic implementation of the layer.

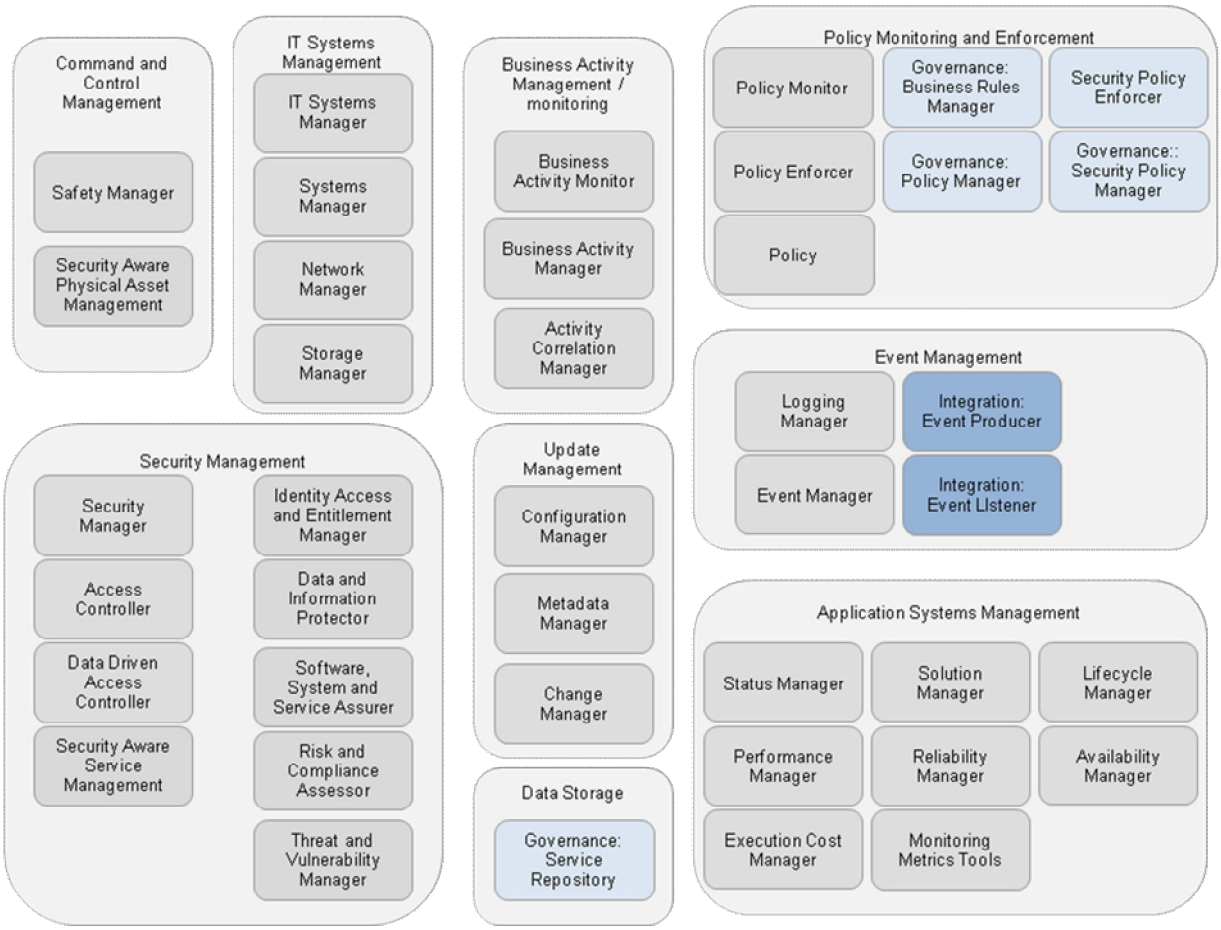


Figure 38 QoS Layer ABB - Part 1

[Add figures to show the interactions and flow; document any particular layer specific groupings of ABBs, as well as mappings of groupings to capabilities and any figures depicting interactions.]

14.3 Quality of Service Layer: Interrelationships between the ABBs

Certain relationships exist between the identified ABBs:
For management of the solution during operation, the ABBs in this diagram interact the following

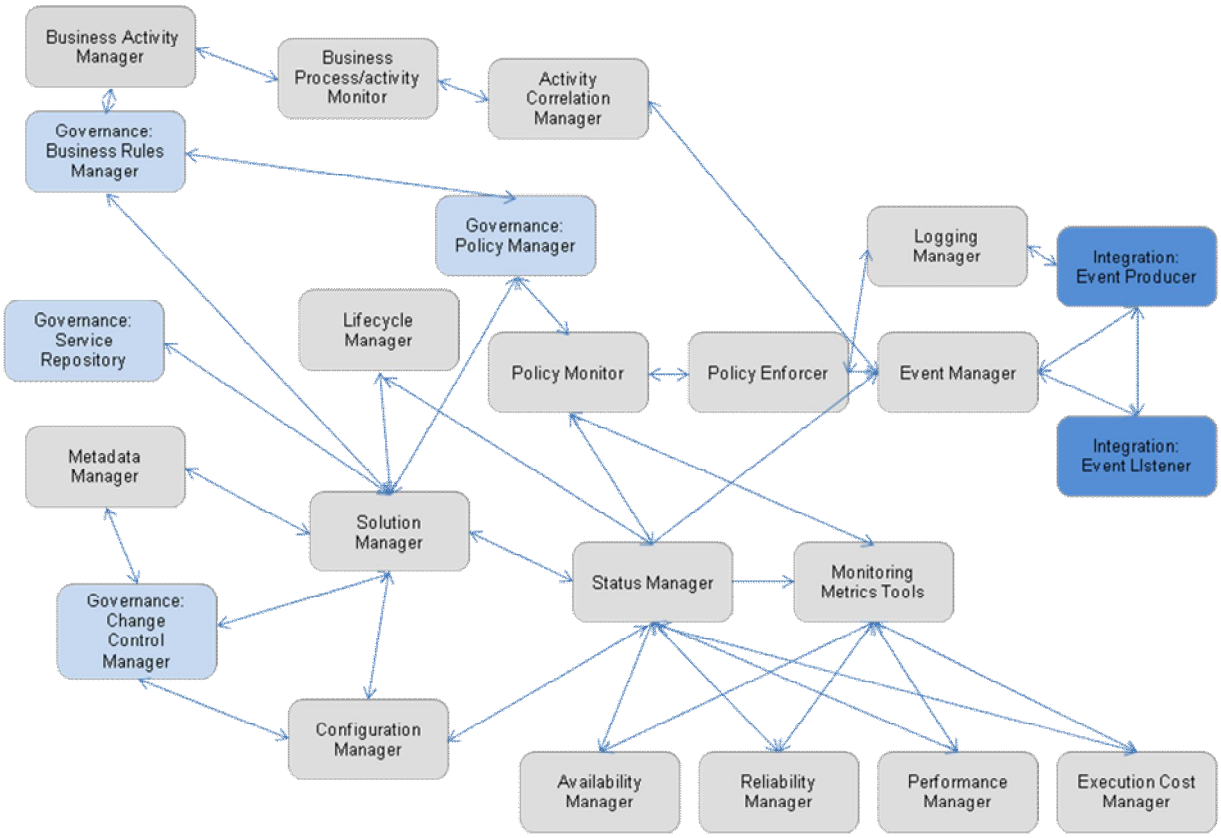


Figure 39. *Component Relationship Diagram – ABBs within the QoS Layer.*

14.4 Quality of Service Layer: Significant Intersection Points with other Layers

14.4.1 Interaction with Other Cross Cutting Layers

The QoS Layer is used by other cross cutting layers to fullfill their respective responsibilities. For example:

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

Certain relationships exist between the ABBs in the QoS layers with other layers:

- An *Solution Manager* ABB interacts with the Consumer layer, the Business Process layer, the Service layer, and the Service Component layer.
- The Solution Manager works with the Repositories and Policy Manager is the Governance Layer.
- The Solution Manger uses the Even producer and Event Listener in the Integration Layer.

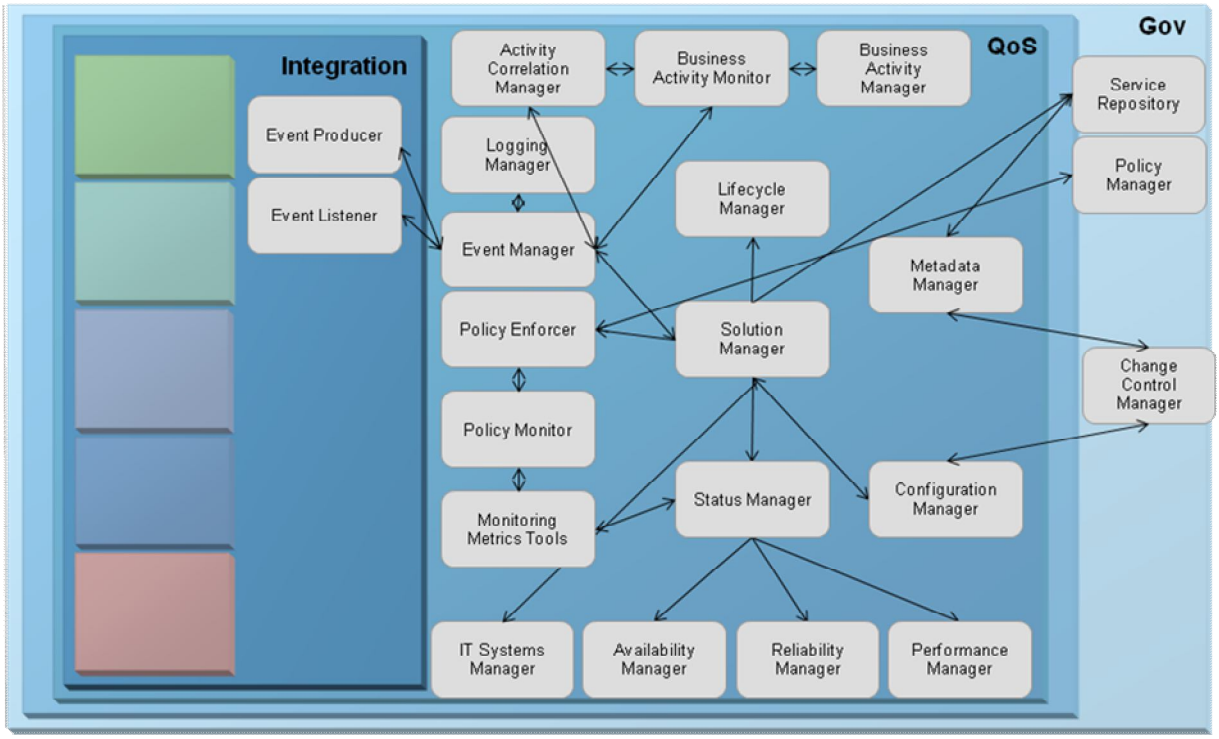


Figure 405 Key Interactions of the QoS Layer with the Other Cross Cutting Layers

The relationship with the governance layer is signific*ant because the Governanc*e layer contains the processes foridentifying and setting the business policies and objectives that generate the QoS non functional requirements.

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 Architecture Layer, and Governance layer. There are other specific usage of the QoS Layer and its ABBs by horizontal layers such as:

1. Policy Enforcer ABB is leveraged by Consumer, Business Process, Service, and Service Component Layers to enforce multitude of policies for the respective layers.
2. Access Controller ABB is leveraged by Consumer, Business Process, Service, and Service Component Layers to enforce security and access control policies for the respective layers.
3. IT systems management manages all the resources in the Operational Systems Layer
4. Status Manager is updated by the Service container when a service changes status.
5. All layers collaborate with the QOS layer via the Solution Manager which coordinates the QOS and security needs of the SOA solution.

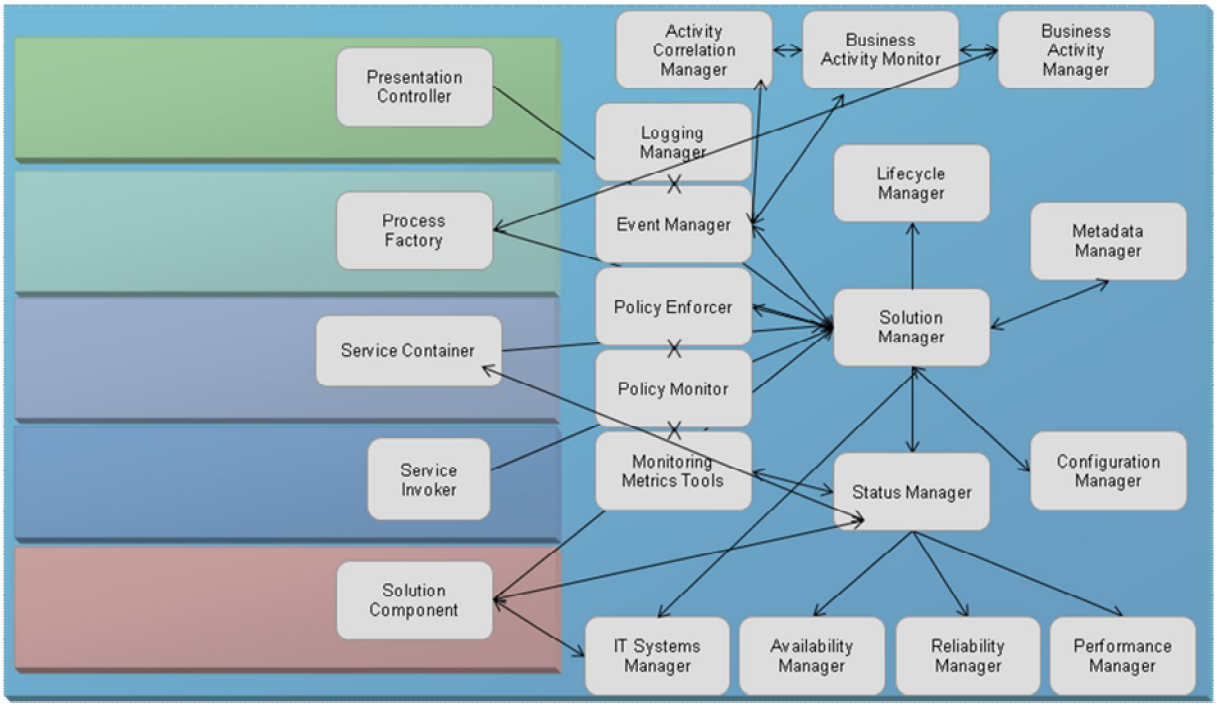


Figure 416 Key Interactions of the QoS Layer with the Horizontal Layers

14.5 Quality of Service Layer: Usage Implications and Guidance

Service components are a managed, governed set of enterprise assets that are funded at the Enterprise or the Business Unit level. In any case, as Enterprise-scale assets they are responsible for ensuring conformance to service level agreements through architectural best practices. This layer typically uses container–based technologies such as application servers to implement the components, has high-availability, load-balancing, etc.

- reflect a service description
- funded, governed, monitored, managed software resource
- abides by the Enterprise Component pattern

The QOS 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 a SOA meets its requirements with respect to e.g.:

- 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.

In particular, SOA security, a significant issue with SOAs' due to their potential perimeter-less nature as opposed to traditional, web-based, "within the firewall", perimeter based security is a capability realized by the QoS layer.

15 Layer 8: Information Architecture Layer

15.1 Information Architecture Layer: Overview

15.1.1 Context and Typical Flow

The Information Architecture 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 this primary objective of this layer are a number of capabilities. This layer includes information architecture, business analytics and intelligence, meta-data 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 meta-data 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:

1. Ability to support information services capability, critical to support a shared, common and consistent expression of data.
2. Ability to integrate information across enterprise in order to enable information services capability.
3. Ability to define meta-data that is used across the SOA RA and in particular the meta-data that is shared across the layers.
4. Ability to secure and protect information.
5. Ability to support business activity monitoring and critical to the usage of the 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 List of Capabilities

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

1. **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, analyze and attempt to improve the quality of data, and ensure consistency and integrity of business-critical data and facts across the enterprise. This approach is known as information as a service approach.

2. **Information Integration**: This category of capabilities addresses the support of information integration and enables capabilities for information services.
3. **Basic Information Management**: This category of capabilities addresses basic information management concerns such as meta-data and unstructured data management.
4. **Information Security and Protection**: This category of capabilities addresses the support of information security and protection concerns.
5. **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 & detect in-time actions.
6. **Information Definition and Modeling**: This category of capabilities defines fundamental constructs of SOA information and events.
7. **Information Repository**: This category of capabilities addresses support of information repository in order to persist data such as meta-data, master data, analytical data, operational data, and un-structured data.

This layer features following capabilities:

Information Services

1. Ability to expose data as services, to add/remove/manipulate data entries in different services or service components, to disable some data from outside access.
2. Ability to interface with the information layer in multiple ways such as message based, service call, batch interface.
3. Ability to handling 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 lifecycle of business entities.
5. Ability to manage 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, transforms the information into the appropriate integrated form, and loads the information into the target repository.
10. Ability to perform Enterprise Information Integration 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 System 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 meta-data in common metadata repository for the enterprise.
17. Ability to manage to capture, aggregate, and manage unstructured content in a variety of formats such as images, text documents, web pages, spreadsheets, presentations, graphics, e-mail, video, and other multimedia.
18. Ability to author, configure, manage, customize and extend meta-data.

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 providing 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 obtains event from the integration layer. Ability to analyze this event information, both in realtime/near-realtime, as well as stored (warehoused) events.
26. Ability to review and assess inbound service activity in the form of event information and determines responses or issues alerts/ notifications

Information Definition and Modeling

27. Ability to define business vocabulary – glossary, terms, business entities.
28. Ability to define 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 re-shaped 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 List of Architectural Building Blocks

The architectural building blocks responsible for providing these sets of capabilities in the Information Architecture 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.		QoS Layer: Event Manager	8
8.	**Information Integration**	Information Integrity Manager	
		Data Cleanser	14
		Data Rationalization Manager	13
		Data Matcher	14
9.		Data Virtualization Manager	
		Data Representation Manager	11
		Data Sourcing Manager	11
10.		Data Cache	15
11.		Integration Layer: Data Transformer	12
12.		Data Consolidator	9
13.		Data Federator	10
14.	**Basic Information Management**	Information Meta-data Manager	16
15.		Content Manager	17
16.		Master Data Authoring Environment	18
17.	**Information Security and Protection**	QoS Layer: Access Controller	19
18.		QoS Layer: Data Driven Access Controller	20

19.		Traceability Enabler/Auditor	21
20.	**Business Analytics**	Data Miner	22
21.		Query, Search, Reporting Engine	23
22.		Analytics Visualization Engine	24
23.		QoS Layer: Business Activity Monitor	25
24.		QoS Layer: Business Activity Manager	25
25.		QoS Layer: Activity Correlation Manager	26
26.	**Information Definition and Modeling**	Business Information - Business Glossary and Terms, Business Entities	27
27.		Common Information – Entities and Data, Messages	28
28.		Business Events	29
29.	**Information Repository**	Data Repository	30 - 32

Table 8 ABB to Capability Mapping for the Information Architecture Layer

15.2 Information Architecture Layer: Details of ABBs and Supported Capabilities

15.2.1 ABBs Details

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
- to the manipulation of data entries in different services and service components; and
- 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 Architecture 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. It provides a consistent entry point to Information Architecture 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 to efficiently join 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 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 Data Validator ABB to validate against business rules.
- invoking Data Quality Manager ABB for enforcing data quality rules
- invoking Event Manager ABB in the QoS 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, relationship such as parent-child relationships and relationships between enterprise data. This ABB is leveraged by 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 both value and structure, and performing data reconciliation including semantic reconciliation. It leverages Information Integrity ABB to fulfill its responsibilities.

15.2.1.7 QoS Layer: Event Manager

See Event Manager ABB in the QoS 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.

15.2.1.8.1 Data Cleanser

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

15.2.1.8.2 Data Rationalization Manger

This ABB is responsible for performing data rationalizing and reconciliation.

15.2.1.8.3 Data Matcher

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

15.2.1.9 Data Virtualization Manager

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

15.2.1.9.1 Data Representation Manager

This ABB is responsible for handling representing 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 the consistency in data.

15.2.1.9.2 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 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 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 he/she needs 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.

15.2.1.10 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.

15.2.1.11 Integration Layer: Data Transformer

See Data Transformer ABB in the Integration Layer.

15.2.1.12 Data Consolidator

This ABB is responsible for extracting relevant information from sources, transforming the information into the appropriate integrated form, and loads the information into the target repository. This ABB supports Extract, Transform, Load, or ETL from one or more source systems into a target system. It is also responsible for support real-time ETL capabilities with the initial or incremental extract, transform, and load of volume data into a target repository (e.g. data warehouse or master data repository).

15.2.1.13 Data Federator

This ABB is responsible for providing Enterprise Information Integration capabilities for federated query access to structured and un-structured data.

15.2.1.14 QoS Layer: Access Controller

See Access Controller ABB in the QoS Layer.

15.2.1.15 QoS Layer: Data Driven Access Controller

See Data Driven Access Controller ABB in QoS Layer.

15.2.1.16 Traceability Enabler/Auditor

This ABB is responsible for monitoring and managing of 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.

15.2.1.17 Information Meta-data Manager

This ABB is responsible to manage and maintain meta-data in common metadata repository for the enterprise, including structured and unstructured data.. It is responsible for managing and maintaining structured and unstructured metadata in a common metadata data repository for the enterprise. For example meta-data 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 SOA RA. It supports, in particular, the ability to store, retrieve and translate meta-data into forms that can be effectively consumed by repositories local to other layers in the SOA RA. It facilitates reuse 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.

15.2.1.18 Content Manager

This ABB is responsible to manage to capture, aggregate, and manage unstructured content in a variety of formats such as images, text documents, web pages, spreadsheets, presentations, graphics, e-mail, video, and other multimedia. It provides the ability to search, catalogue, secure, manage, and store unstructured content to support the creation, revision, approval, and publishing of content. It provides the ability to identify new categories of content and create taxonomies for classifying enterprise content. This ABB is also responsible to manage 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.

15.2.1.19 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.

15.2.1.20 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 to build 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.

15.2.1.21 Query, Search, Reporting Engine

This ABB is responsible for supporting ad hoc queries, search, reporting, slicing/dicing/ drill downs, online analytical processing (OLAP) capabilities for enterprise information.

15.2.1.22 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 presentations 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.

15.2.1.23 QoS Layer: Business Activity Monitor

See Business Activity Monitor ABB in QoS Layer.

15.2.1.24 QoS Layer: Business Activity Manager

See Business Activity Manager ABB in QoS Layer.

15.2.1.25 QoS Layer: Activity Correlation Manager

See Activity Correlation Manager ABB in QoS Layer.

15.2.1.26 Business Information

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

15.2.1.27 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 service oriented paradigm.

15.2.1.28 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 service oriented paradigm.

15.2.1.29 Data Repository

This ABB provides the essential foundation for the storage of operational and re-shaped information that adds business value. The core data repositories are Analytical Data, Operational Data. Master Data, Un-structure Data, Meta-Data.

- **Analytical Data Repository**: Includes operational data stores, data warehouse, data mart, staging areas, and adhoc 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.
- **Un-structured Data Repository**: Includes textual object, graphical objects, multi-media objects etc.
- **QoS Layer: Meta-Data Repository**: Enables storage of meta-data. It stores meta-data 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 Information Architecture Layer can be thought of as being logically partitioned into following categories which support:

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

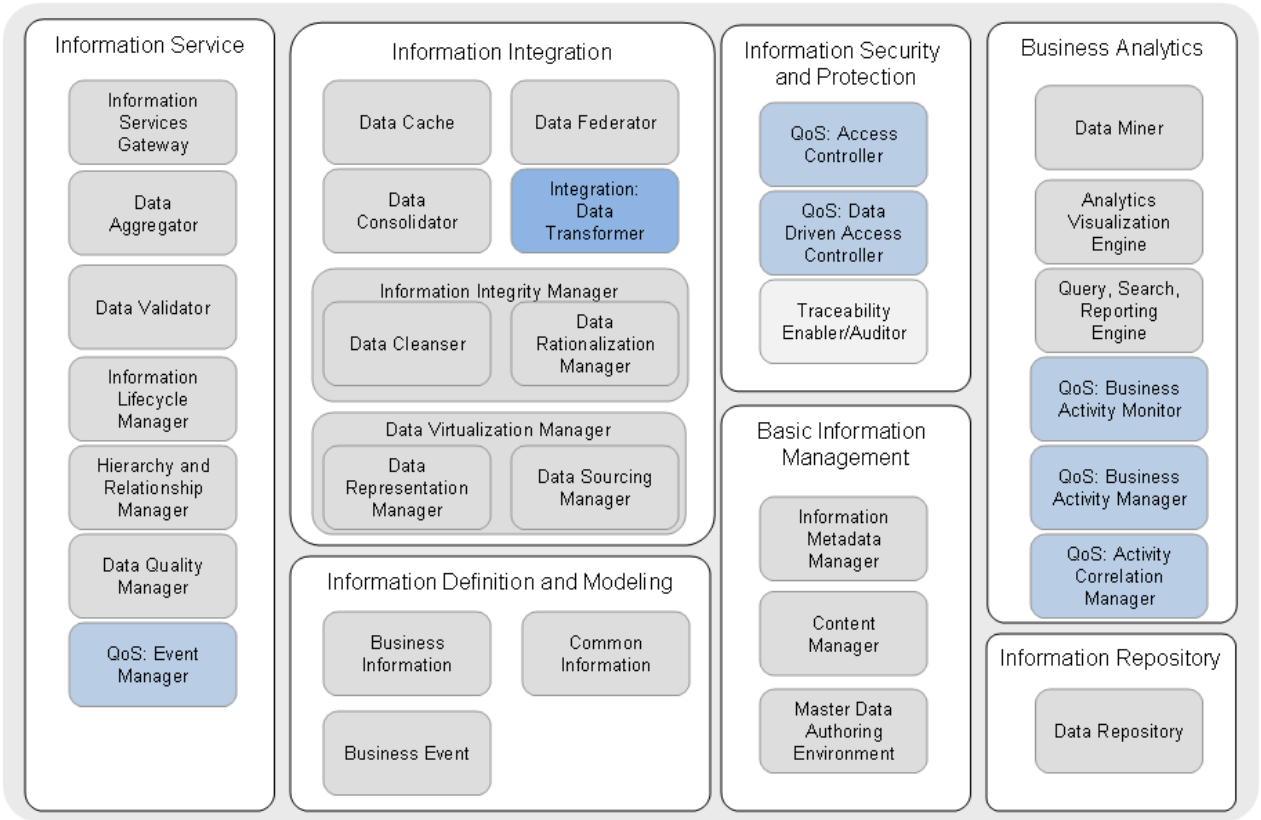


Figure 42 ABBs in the Information Architecture Layer

15.3 Information Architecture Layer: Interrelationships between the ABBs

The relationship among these ABBs is shown for different scenarios.

The first scenario is for information as a service where information retrieved from multiple sources.

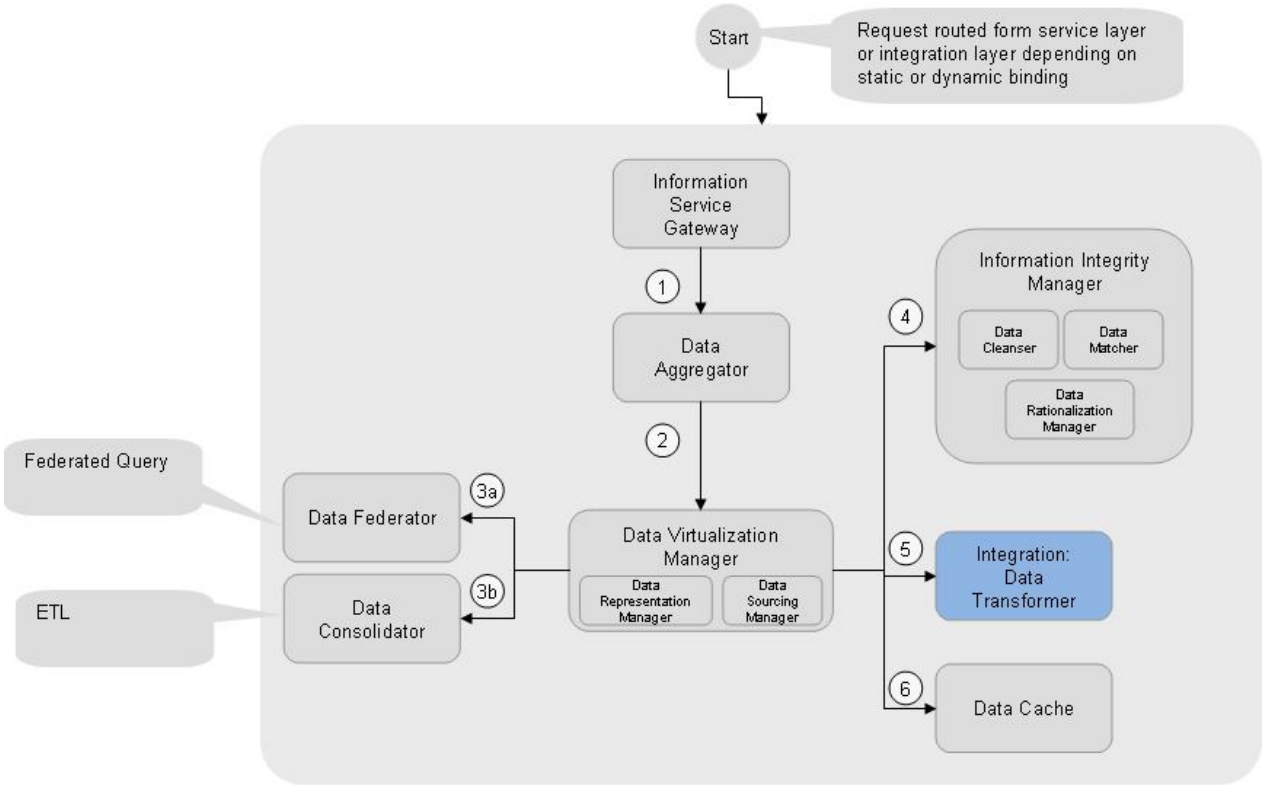


Figure 43 Key Interactions among ABBs in the Integration Layer in an IaaS Query Scenario

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

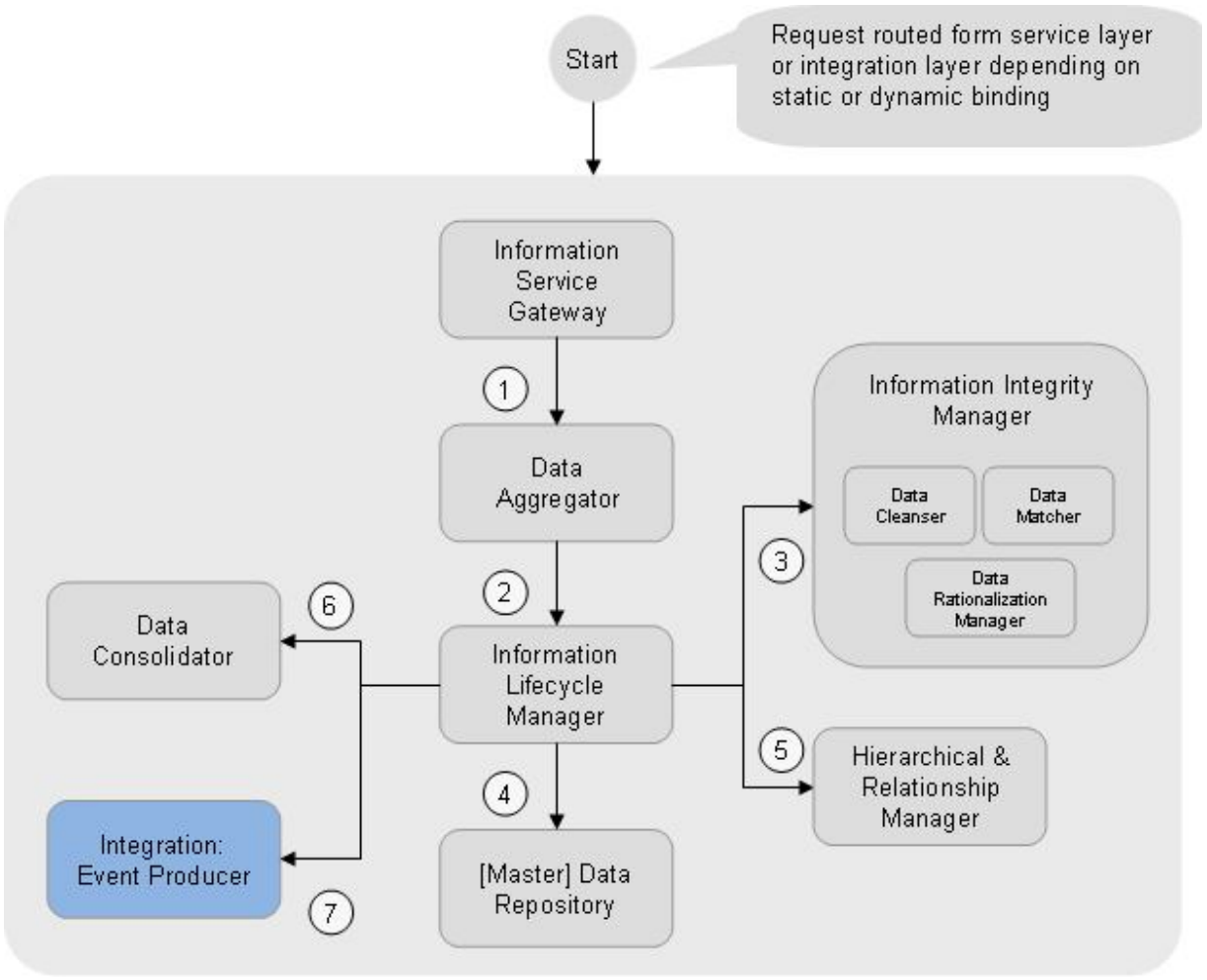


Figure 44 Key Interactions among ABBs in the Integration Layer for an Add/Update in an MDM scenario

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

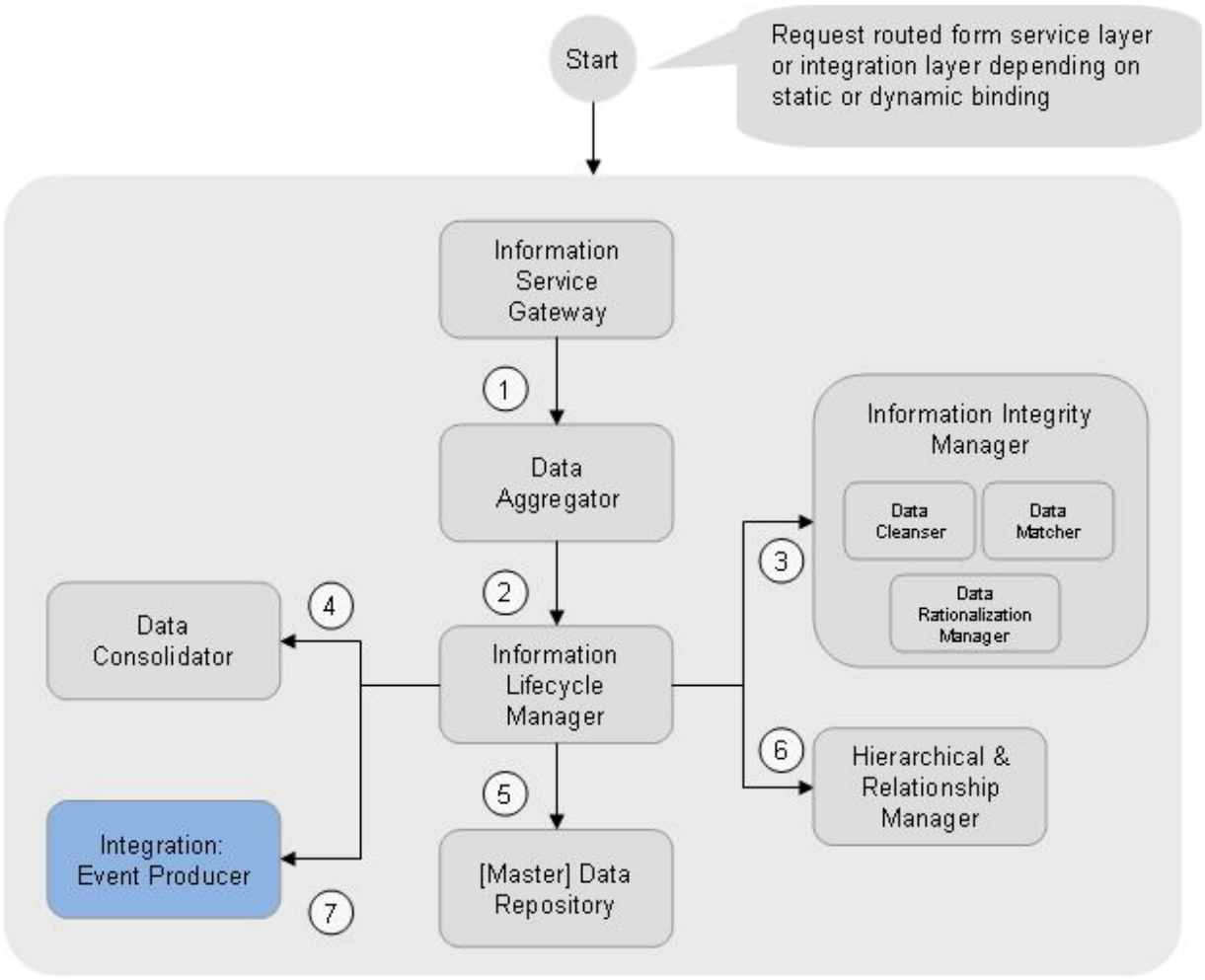


Figure 45 Key Interactions among ABBs in the Integration Layer for a Delta Extract and Update in an MDM scenario

15.4 Information Architecture Layer: Significant Intersection Points with other Layers

Certain relationships exist between the ABBs in the Information Architecture Layer with those in other cross cutting and horizontal layers:

- Information Service Gateway ABB interacts with the Consumer Layer, the Business Process Layer, the Service Layer, the Service Component Layer, the Operational System Layer, the Integration Layer, the QoS Layer, and the Governance Layer.
- Traceability Enabler/Auditor ABB interacts with the QoS Layer.

- Data Sourcing Manager ABB interacts with the Operational Systems Layer and the Governance Layer.

15.4.1 Interaction with Other Cross Cutting Layers

1. This layer leverages Event Manager ABB in the QoS 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.
2. This layer leverages Data Transformer ABB in the Integration Layer for transforming and mapping of data from one format to another and aligning data from different resources.
3. This layer leverages Access Controller ABB in QoS Layer to enforce security policies and access privileges.
4. This layer leverages Data Driven Access Controller ABB in QoS Layer to enforce access privileges on individual data items.
5. This layer leverages Business Activity Monitor ABB, Business Activity Manager ABB, and Activity Correlation Manager ABB in the QoS Layer to monitor events, business activities, 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.
6. Policy Enforcer ABB in this layer is leveraged by 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 Services Layer to enforce service policies, and by Business Process Layer to enforce of polices on business processes.

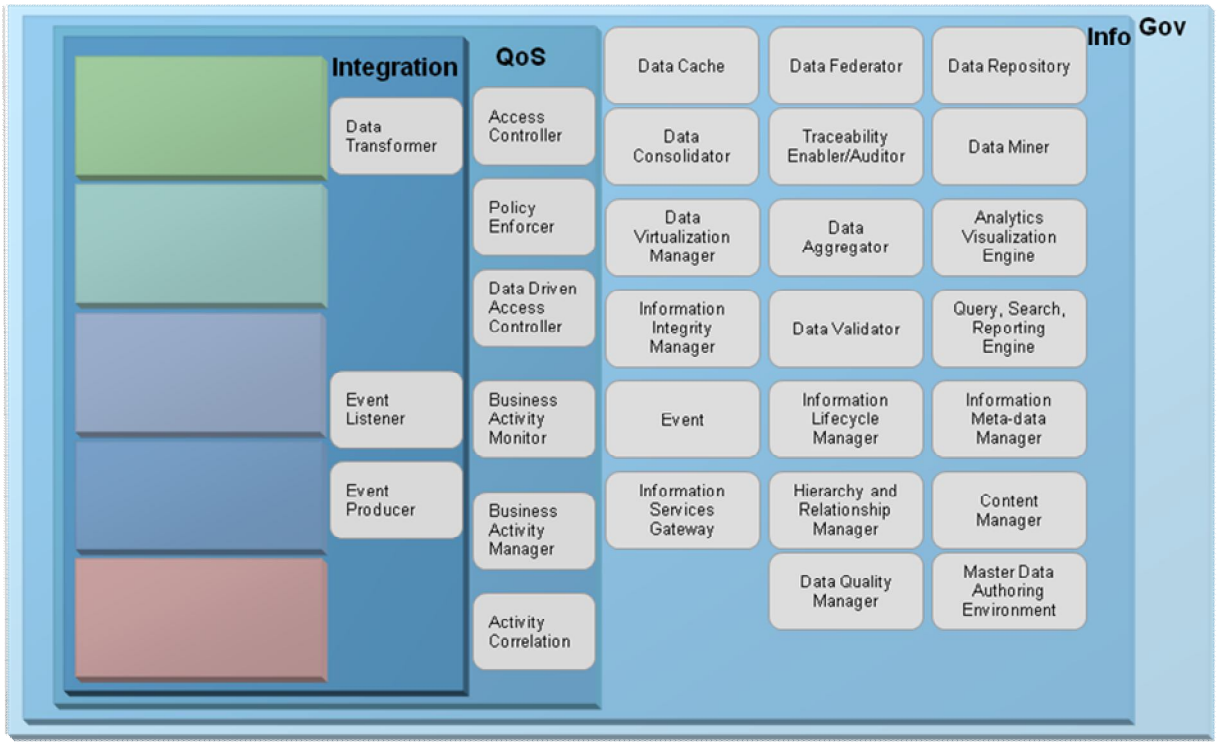


Figure 46 Key Interactions of the Information Architecture Layer with the Other Cross Cutting Layers

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, Service Layer and Service Component Layer require information (structure and unstructured data, meta-data and messages) to fulfill their respective responsibilities and therefore, rely on the Information Architecture Layer to access information. These horizontal layers are dependent on the ABBs of the Information Architecture Layer to fulfill their information needs.

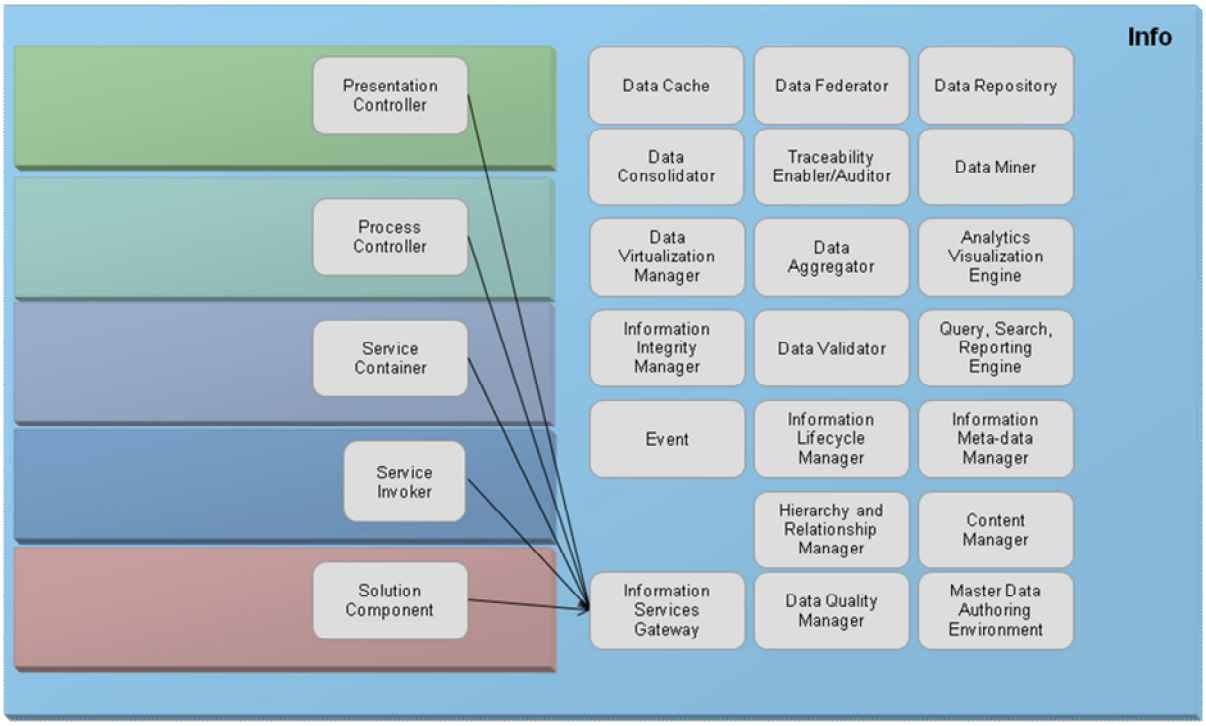


Figure 47 Key Interactions of the Information Architecture Layer with the Horizontal Layers

15.5 Information Architecture Layer: Usage Implications and Guidance

Especially, for industry-specific SOA solutions, this layer captures all the common cross-industry and industry-specific data structure, XML-based meta-data Architectures (e.g. XML schema), and business protocols of exchanging business data. Some discovery, data mining, and analytic modelling of data are also covered in this layer. These common structures may be standardized for the industry or organization.

16 Layer 9: Governance Layer

16.1 Governance Layer: 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 & 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 life-cycle). 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. SLA, capacity and performance, security and monitoring).

This layer can be applied to all the other layers in the SOA Reference Architecture. From quality of service and management perspective, it is well connected with QOS Layer. From a service lifecycle and design time perspective it is connected with the Service Layer. From a 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:

- Service Level Agreements based on QoS and 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 enterprise architecture and a solution architecture view.

16.1.1 Context and Typical Flow

This layer features the following characteristics:

- Defines policies, compliance and exceptions characteristics
- Monitors the health of SOA services, solution and governance
- Reports on compliance, exceptions, service health, versions
- Provides a consolidation point for business rules

The Governance Layer is aligned with the standardized SOA Governance Framework (https://www.opengroup.org/soa/standards/gov.htm#gov) which defines a 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 fulfil the responsibilities of the Governance Layer:

- Compliance processes are the conformance processes and policies including vitality which ensures the continued relevance of key governance model constructs, such as reference models, life-cycle (activities, work products), etc.
- Dispensation processes are processes and policies to handle exception to compliances
- Communication processes disseminate and educate the fundamental constructs of life-cycle, 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 life-cycle, 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 it relates to the modeling of SOA solutions throughout the life-cycle, from identification to retirement.

- Service portfolio management processes – describe activities for selecting services to be developed and services to be reused 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.

- Plan governance phase is responsible for analyzing, arranging, and scheduling solution-level monitoring and management
- Define governance phase is responsible for defining a solution-specific SOA-based governance control model and strategy
- Implement governance phase is responsible for enabling and realizing solution-level governance control
- 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 architectural 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 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 development and configuration 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 List of Capabilities

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

1. **Governance Planning**: This category of capabilities provides ability to plan governance.
2. **Governance Definition:** This category of capabilities provides ability to define governance.
3. **Governance Enablement and Implementation:** This category of capabilities provides ability to implement governance.

4. **SOA Meta-data Storage and Management:** This category of capabilities enables storing and managing service metadata and governance artifacts.
5. **Business Rule Definition and Management:** This category of capabilities provides ability to define and manage business rules.
6. **Policy Definition and Management:** This category of capabilities provides ability to define and manage policies.
7. **Monitoring**: This category of capabilities provides ability to monitor application of policies, governance processes, and effectiveness of governance.
8. **Management:** This category of capabilities provides ability to manage governance artifacts and processes.
9. **Workflow:** This category of capabilities provides ability to capture and automate governance processes.

This layer features 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 Meta-Data Storage and Management

7. Ability to store and search any kind of assets.
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:
    i. Service contract definition (e.g. WSDL)
    ii. Service policy management
    iii. Service version information management
    iv. Service dependencies (e.g. the ability to integrate with a CMDB tool)
    v. Service management descriptions
    vi. Canonical form and domain model specification for integration with the Information Architecture 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 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 List of Architectural Building Blocks

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 Meta-data Storage and Management**	Asset Repository	7
		Service Repository	8 - 14
2.		Service Registry	13, 14
3.	**Business Rule Definition and Management**	Business Rule	15
4.		Business Rules Manager	15 - 17
5.		Business Rules Repository	17
6.	**Policy Definition and Management**	Policy	18
7.		Policy Manager	18 - 21
8.		QoS Layer: Policy Monitor	22
9.	**Monitoring**	QoS Layer: Monitor Metrics Tools	22 - 27
10.		Dashboard	22, 23, 28, 29
11.	**Management**	Reporting Tools	29
12.		Development Tools	3 - 6
13.		QoS Layer: Configuration Manager	30
14.		Change Control Manager	31
15.		QoS Layer: Access Controller	32
16.		Governance Gateway	30 - 32
17.	**Workflow**	Workflow Process	33 - 34

Table 9 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 Governance Layer: 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 ABBs Details

16.2.1.1 Asset Repository

This ABB enables organizations to organize, govern, and manage their valuable assets scattered through out the enterprise and to encourage reuse of existing assets within the organization. It provides 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 Reusable Asset Specification (RAS) from OMG.

16.2.1.1.1 Service Repository

This ABB is a special kind of Asset Repository ABB and primarily focuses on services and related artifacts in SOA and artifacts governing the SOA.

It acts as a design time repository to store and locate metadata about services, including descriptions of the service contract, information about the quality of service 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, run-time operational aspect, and life cycle 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.

16.2.1.2 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 quality of service policies and security, versioning information, and runtime information such as end-points. This ABB may leverage the design time Service Repository ABB to fetch meta-data about services to fulfill the runtime needs of SOA such as dynamic/runtime binding and service virtualization. Standards for registries include UDDI.

Since 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 where changes to say the location of a service (expressed in terms of an “endpoint” which is the location from which a service is invoked, which for example may be the service container’s address or some other unique identifier (URI)).

16.2.1.3 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 service oriented paradigm. The business rules may apply may apply throughout the SOA solution and governance lifecycle.

16.2.1.4 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.

16.2.1.5 Business Rules Repository

This ABB is responsible for storing and accessing business rules artifacts. This ABB is used by the Business Rule Manager ABB.

16.2.1.6 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 service oriented paradigm. Policies may apply throughout the SOA solution and governance lifecycle. Policies could be at multiples level 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 QoS Layer.

16.2.1.7 Policy Manager

This ABB is responsible for defining, authoring, distributing, and maintaining policies using policy management tools. Sophisticated policy manager 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 represented by the Policy Enforcer ABB in the QoS Layer and its intersection with the other layers of SOA RA for evaluation and enforcement purposes.

16.2.1.8 QoS Layer: Policy Monitor

See Policy Monitor ABB in the QoS Layer.

16.2.1.9 QoS Layer: Monitoring Metric Tools

See Monitoring Metric Tools ABB in the QoS Layer.

16.2.1.10 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 Monitoring Metric Tools ABB and Policy Enforcer ABB in the QoS Layer.

16.2.1.11 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.

16.2.1.12 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.

16.2.1.13 QoS Layer: Configuration Manager

See Configuration Manager ABB in the QoS Layer.

16.2.1.14 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.

16.2.1.15 QoS Layer: Access Controller

See Access Controller ABB in the QoS Layer.

16.2.1.16 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.

16.2.1.17 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 done. Standards include BPEL.

16.2.2 Structural Overview of the Layer

The Governance Layer applies to all of the other layers of the SOA RA. ABBs of the Governance Layer can be thought of as being logically partitioned into categories that support:

1. Ability to store and access, (ie with registries, repositories, web sites, or databases) artifacts related to governance. Artifacts can be organization documentation, business process documentation, policies, compliance records, etc.
2. Ability to define and manage business rules.
3. 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.
4. 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.

5. Ability to manage and maintain governance, including the ability to do configuration, change control, security , and reporting
6. Ability to automate processes and capture the processes in workflows.

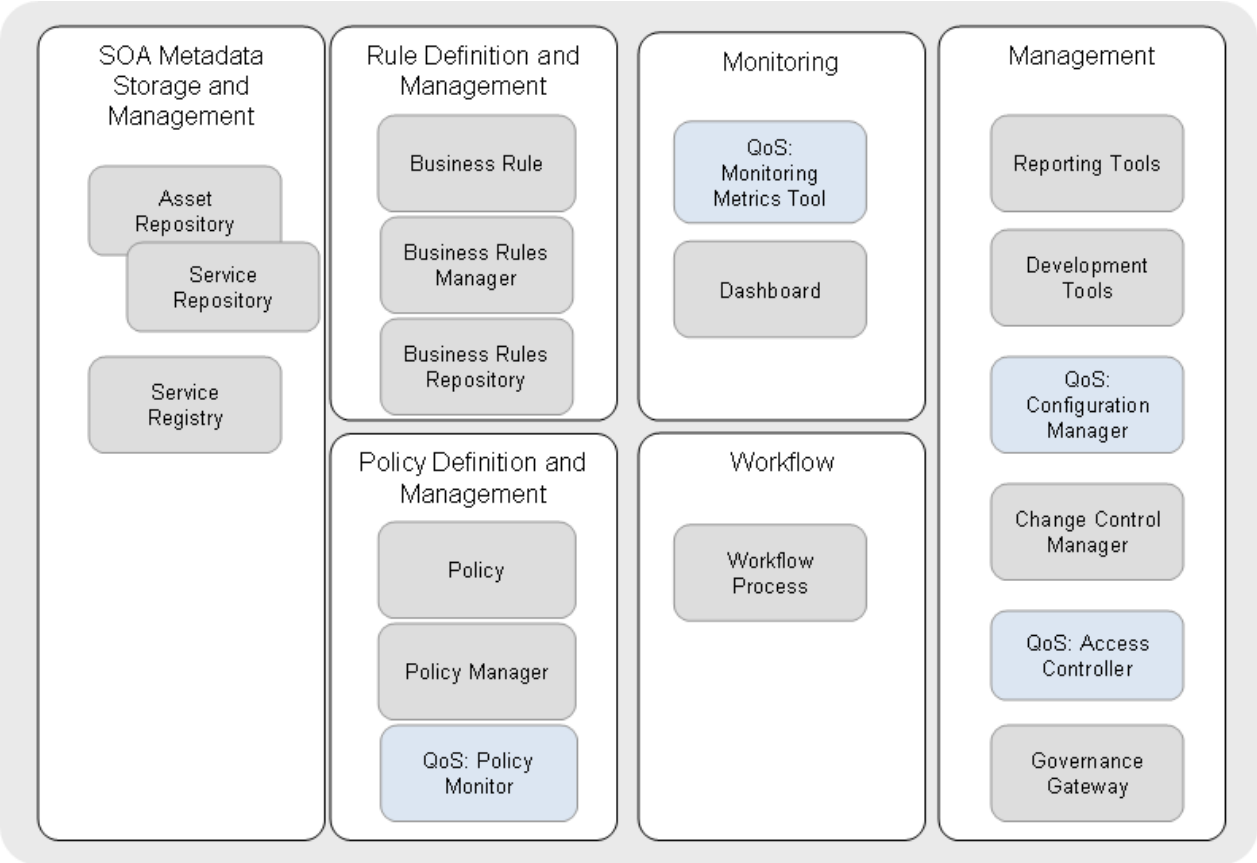


Figure 48 ABBs in the Governance Layer

Governance Layer leverages ABBs from QoS Layer to fulfil its core responsibilities. The ABBs from QoS Layer leveraged by the Governance Layer are: Poliy 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 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, 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 Service Registry ABB for governance information for services, Service Repository ABB for services and governance metadata, Policy Manager ABB to author policies and QoS Layer: Configuration Manager to configure the QoS Layer: Policy Enforcer ABB; QoS 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 QoS Layer: Policy Enforcer ABB, QoS 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 Governance Layer: Interrelationships between the ABBs

The figure below illustrates the different ABBs and their interdependencies.

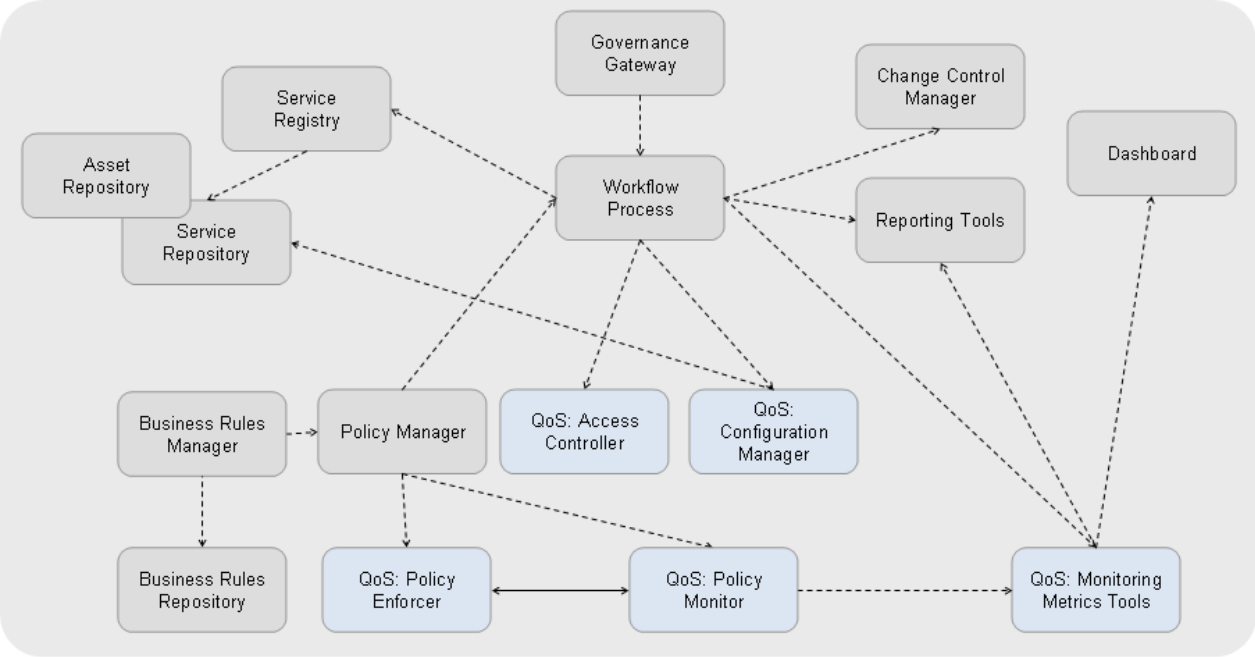


Figure 49 Relationships among ABBs in the Governance Layer

The Governance Gateway ABB invokes and governs the governing processes captured as workflows in 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 QoS 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 QoS 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 QoS Layer, Access Controller ABB in the QoS Layer, and governing process workflows.

The Policy Enforcer ABB in the QoS Layer and governing process Workflow ABB shares policy results with the Monitoring Metrics Tool ABB in the QoS Layer.

Here is a flow for an example compliance process

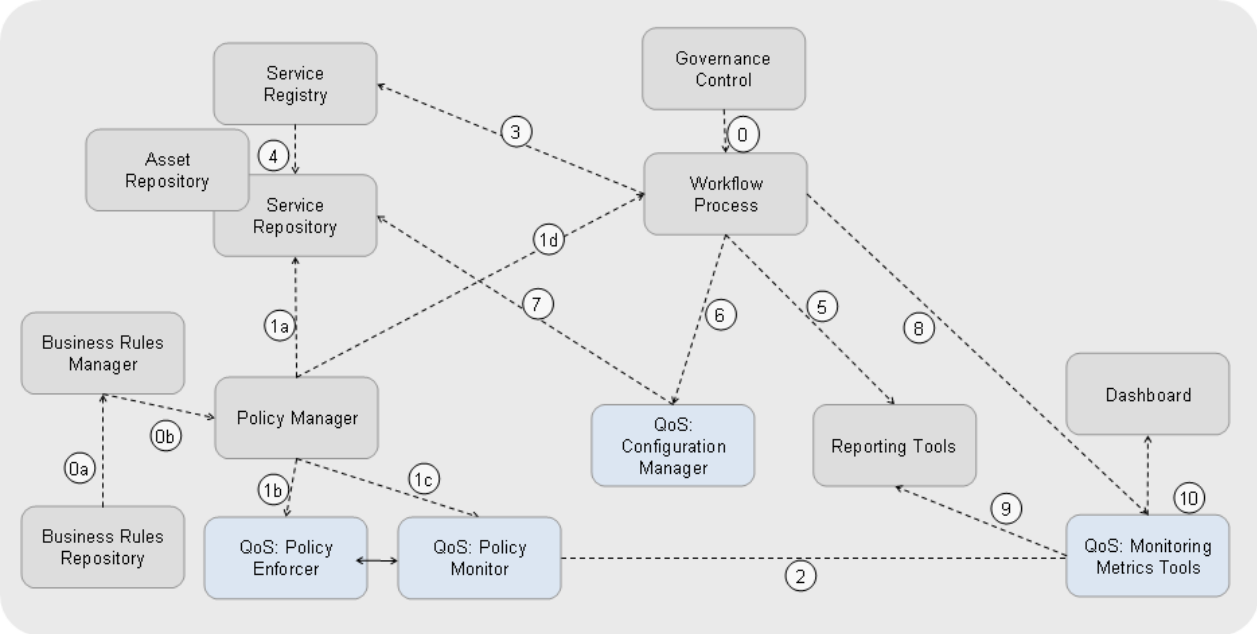


Figure 50 Sample Interactions among ABBs in the Governance Layer for a Governance Compliance Process

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. At (1a-d) the policy to inactivate a service after 5 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 5 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 QoS 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 updates the dashboard with a new status for the service being out of use. Here it is clear that the governance and QOS layers work cooperatively.

16.4 Governance Layer: 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 QoS, Integration, Services, and Business Process layers contain ABBs that the governance layer needs to leverage.

16.4.1 Interaction with Other Cross Cutting Layers

The Governance Layer is responsible for governing all the other cross cutting layers, namely, Integration, Information, and QoS. In addition, the Governance Layer relies on ABBs in the QoS to fulfil its core responsibilities.

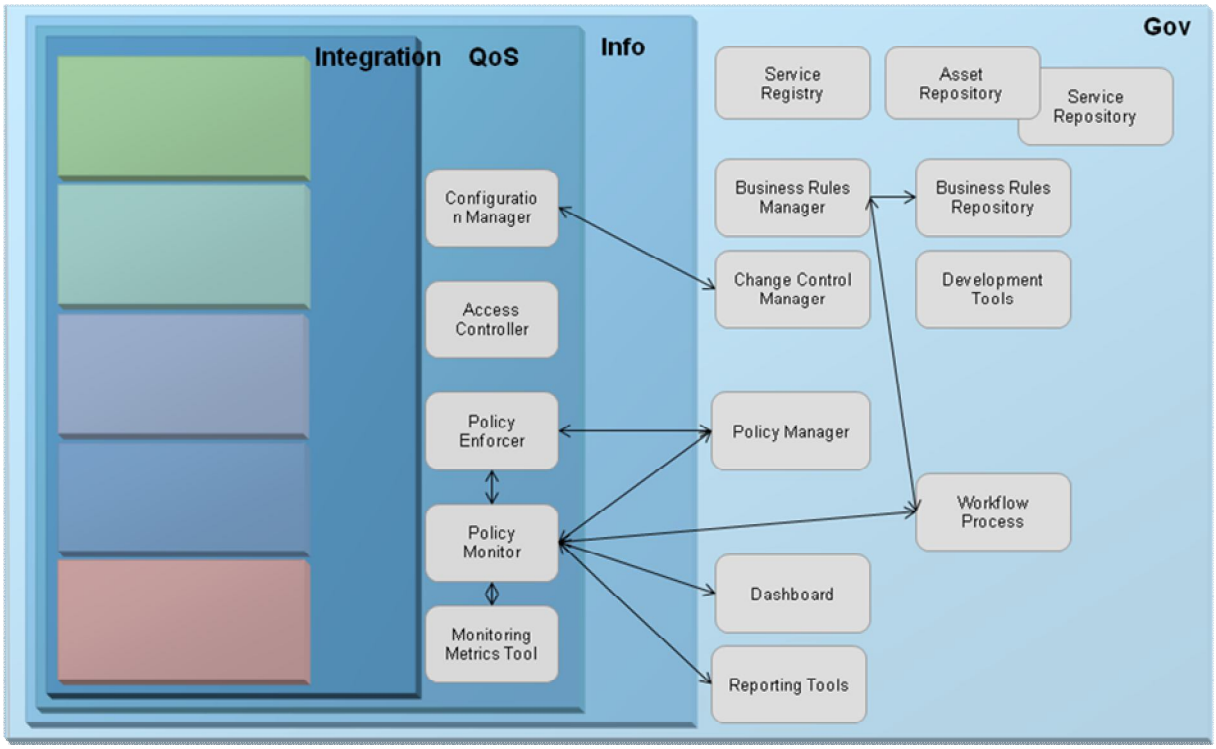


Figure 51 Key Interactions of the Governance Layer with the Other Cross Cutting Layers

Governance defines the policies used to drive the aspects of quality-of-service in the QoS layer. The governance layer is dependent on the QoS layer to provide the following capabilities:

a. Business Rule Manager ABB is needed to ensure the appropriate policies are set to support the policy manager capability. It also supports the Policy Enforcer ABB in the QoS Layer in fulfilling its responsibilities through the Policy Manager ABB.
b. It leverages the Policy Enforcer ABB in the QoS Layer to enforce policies related to change control processes as required by Change Control Manager ABB to ensure change control is performed appropriately. Similarly, it leverages the Policy Enforcer ABB in the QoS Layer to enforce security policies and policy to configure the solution using the Configuration Manager ABB in the QoS Layer. It also leverages

Policy Enforcer ABB in the QoS Layer to enforce governance policy and monitoring metrics for a solution.

c. It leverages Access Controller ABB in QoS he governance layer defines the policies used to configure the security for the SOA solution and the governing processes via the Security manager capability.
d. It leverages Configuration Manager ABB in the QoS 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 QoS Layer. In the case there are automated governing process workflows, the workflow adjusts and changes the configuration using the Configuration Manager ABB in the QoS Layer in order to adhere to the governance policies.
e. It leverages Monitoring Metrics Tools ABB and Policy Enforcer ABB in the QoS 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. Dashboard ABB leverages Monitoring Metrics Tools ABB in QoS 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:

1. IT infrastructure and implementation assets are governed in Operational Systems Layer.
2. Enterprise component are governed in Service Component Layer.
3. Services are governed in Service Layer. Policies defined by governance decide which services will be built and reused.
4. Business processes are governed in Business Process Layer.
5. Governance and Integration intersection – policies defined by governance govern the mediation of interactions. 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 kind of policies by interfacing with Policy Manager ABB. In addition to governing these horizontal layers, the Governance Layer uses Business Process Layer to capture governance process and Workflow ABB leverages Business Process Layer to define the governance processes.

Service Layer leverages Service Repository ABB to store service definition/contracts, policy and meta-data about service during design time. Service Layer leverages Service Registry ABB to store

service definition/contracts and policy and meta-data about service to be used during run time in order to discover services and bind to service provider/end-point enabling service virtualization.

Integration Layer leverages Service Registry ABB to determine end-point for a service request and to enable service virtualization.

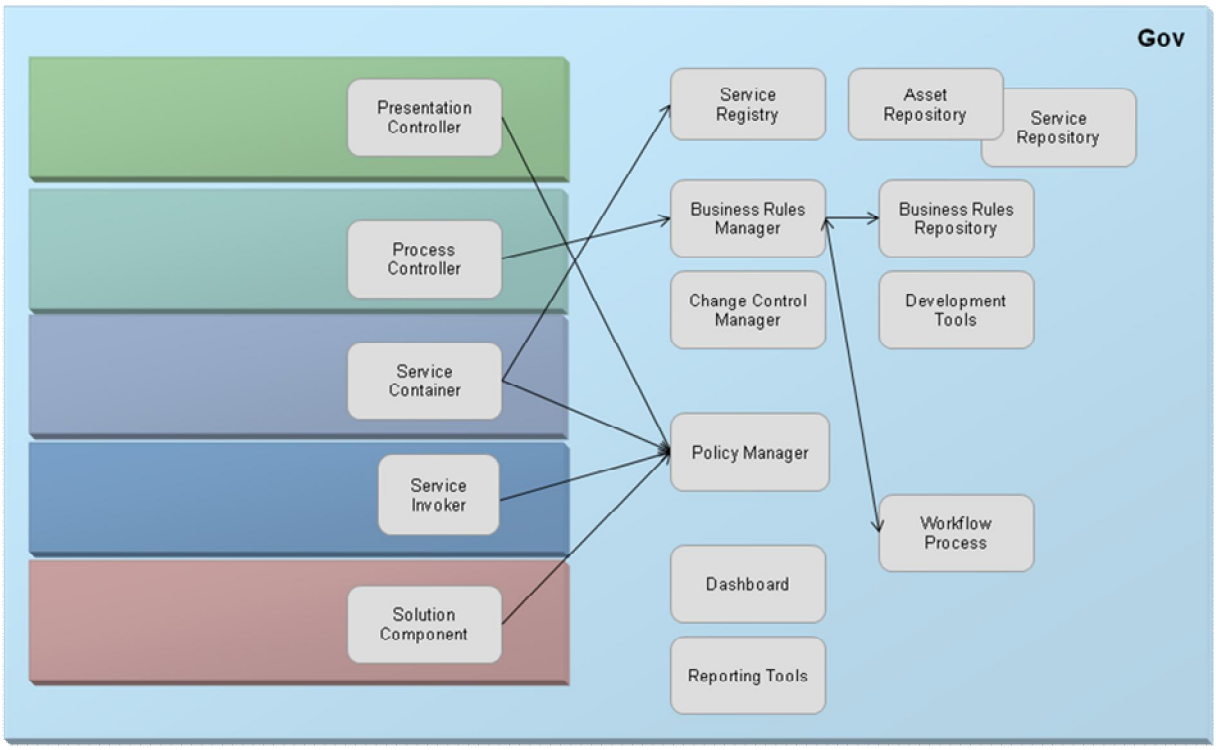


Figure 52 Key Interactions of the Governance Layer with the Horizontal Layers

In summary, Governance ABBs are used by other SOA RA layers:

1. The Service Repository ABB is used by Service, QoS, and Integration layers.
2. The Service Registry ABB is used by the Service, Business Process, Consumer, Integration, and QoS layers.
3. The Policy Manager ABB is used by the Integration and QOS layers.
4. The Business Rule Manager ABB is used by the Business Process, Services, Service Component and QoS layers.

16.5 Governance Layer: 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 life-cycle 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 e.g.

- 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.

Other important responsibility of Governance Layer is to measure, gather, evaluate, and test metrics against policies on a regular basis. Key performance indicators 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 Reference Architecture

The SOA RA provides a mechanism for use in a variety of scenarios:

1. For organizations adopting SOA
2. For organizations building SOA components (SOA product vendors)
3. For organizations providing services in the construction of SOAs (integrators)
4. 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 Reference Architecture 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,
- and legacy componentization and transformation.

As we apply an SOA Modelling and Delivery Methods every element of SOA that is identified is mapped back to the SOA Reference Architecture providing a “dashboard view” of the SOA in progress; useful as a communication means for various business and IT stakeholders.

In addition, the SOA Reference Architecture is used to define capabilities and the technical feasibility of solutions. This usage is focused on as 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 Reference Architecture provides a checklist of key elements that must be considered when you build your SOA: mandatory as well as optional layers, attributes, architectural building blocks, 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 are an important part of building SOA solutions. In this specification, we just focus for the most part on the areas that are critical success factors in building service-oriented architectures.

The SOA Reference Architecture applies to various types of practitioners such as Enterprise Architects, Solution Architects, etc. The SOA Reference Architecture is an abstract, logical design of a SOA. Thus it answers the question of "What is a SOA?" Architects can use it as a checklist of layers, architectural building blocks and their relations in each layer, the options available, decisions that need to be made at each layer. The layers provide a starting point for the separation of concerns needed to build a 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 above reference architecture, to a given level of granularity of a SOA Eco-system, we will typically find the need to create the same layers for each level of granularity. Thus, Enterprise Architecture might use the SOA Reference Architecture 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 a SOA or Services Eco-system, a company would need to have a standard reference Architecture such as that depicted by the SOA Reference Architecture, 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 Appendix –

Glossary

Note to Reviewers:

For like terms, this glossary will use the Open Group SOA Ontology and the glossaries of other Open Group SOA working groups.

Term	Definition
Capability	<Open Group Definition>; in the context of this document <extension for this document>

References

[1] Ali Arsanjani, Service-oriented modeling and architecture:How to Identify, Specify and Realize your Services, IBM developerWorks, (http://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, http://www.ibm.com/developerworks/library/ar-archtemp/

[3] 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 Page(s):10 - 17"

[4] Liang-Jie Zhang, 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, Marcia Stockton, SOA Programming Model, http://www-128.ibm.com/developerworks/webservices/library/ws-soa-progmodel/

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

[7] D. L. Parnas, P. C. Clements, and D. M. Weiss. The modular structure of complex systems. *IEEE Transactions on Software Engineering*, SE-11(3), 1985, pp. 259-266.

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

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

[10] Gordon Bell, Allen Newell, Daniel Sieworek, Computer Structures: Principles and Examples, McGrawHill, 1982, Chp 3

[11] Enterprise Continuum, The Open Group Architecture Framework Version 9, , https://www.opengroup.org/architecture/togaf9-doc/arch/toc-pt5.html, TOGAF Online > Part V: Enterprise Continuum and Tools > Enterprise Continuum, 2009

[12] Definition of SOA 1.1, The Open Group SOA Work Group, https://www.opengroup.org/projects/soa/doc.tpl?gdid=10632, June 8, 2006

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

[14] Architecture Building Block Definition, The Open Group Architecture Framework Version 9, http://www.opengroup.org/architecture/togaf9-doc/arch/chap03.html, Definitions, 2009

[15] OASIS Open Composite Services Architecture (Open CSA) Member section, OASIS, http://www.oasis-opencsa.org/

[16] Web Services for Remote Portlets (WSRP) V1.0, OASIS, August 2003, http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf

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

[18] Voice Extensible Markup Language (VoiceXML), W3C Recommendation, March 2004, http://www.w3.org/TR/voicexml20/

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

[20] Microsoft .net Framework, Microsoft, Redmond, WA, http://www.microsoft.com/NET

[21] XML Transformations (XSLT) version 1.0, W3C Recommendation November 1999, http://www.w3.org/TR/xslt

[22] Garrett, JJ, (2005-02-18). « Ajax : a New Approach to Web Applications ». Adaptive Path.com. . Retrieved on 2008-06-19

Index

Error! No index entries found.