INTERNATIONAL STANDARD

ISO/IEC 18384-2

First edition
2016-07-01

Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) —

Part 2: Reference Architecture for SOA Solutions

Technologie de l'information — Architecture de référence pour l'architecture orientée service (SOA RA) —

Partie 2: Architecture de référence pour les solutions de l'architecture orientée service

Reference number
ISO/IEC 18384-2:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 38, *Cloud Computing and Distributed Platforms.*

ISO/IEC 18384 consists of the following parts, under the general title *Information Technology — Reference Architecture for Service Oriented Architecture (SOA RA)*:

— *Part 1: Terminology and concepts for SOA*

— *Part 2: Reference Architecture for SOA Solutions*

— *Part 3: Service Oriented Architecture Ontology*

Introduction

Service oriented architecture (SOA) is an architectural style in which business and IT systems are designed in terms of services available at an interface and the outcomes of these services. A service (see ISO/IEC 18384-1:2016, 3.20) is a logical representation of a set of activities that has specified outcomes, is self-contained and may be composed of other services but consumers of the service need not be aware of any internal structure.

SOA uses services to create and integrate information systems so that they are suitable for a variety of business and application requirements. SOA enables interactions between businesses without needing to specify specifics of any particular business domain. Using the SOA architectural style can improve the efficiency of developing information systems and integrating and reusing IT resources. In addition, using the SOA architectural style can help enable rapid response of information systems to ever-changing business needs.

ISO/IEC 18384 is intended to be a single set of SOA technical principles, specific norms, and standards for the world-wide market to help remove confusion about SOA and improve the standardization and quality of solutions.

ISO/IEC 18384 defines the terminology, technical principles, reference architecture, standard service categories and ontology for SOA. This part of ISO/IEC 18384 can be used to introduce SOA concepts, as a guide to the development and management of SOA solutions, as well as be referenced by business and industry standards.

ISO/IEC 18384 contains three parts:

a) ISO/IEC 18384-1, which defines the terminology, basic technical principles and concepts for SOA;

b) ISO/IEC 18384-2, which defines the detailed SOA reference architecture layers, including a metamodel, capabilities, architectural building blocks, as well as a set of categories or types of services in SOA solutions;

c) ISO/IEC 18384-3, which defines the core concepts of SOA and their relationships in an ontology.

The targeted audience of ISO/IEC 18384 includes, but is not limited to, standards organizations, architects, architecture methodologists, system and software designers, business people, SOA service providers, SOA solution and service developers, and SOA service consumers who are interested in adopting and developing SOA.

Users of this part of ISO/IEC 18384 will find it useful to read ISO/IEC 18384-1 for an understanding of SOA basics. ISO/IEC 18384-1 should be read before reading or applying this part of ISO/IEC 18384. For those new to the SOA reference architecture, Clause 4 provides a high-level understanding of the Reference Architecture for SOA Solutions. The remaining clauses provide comprehensive details of the architectural building blocks and trade-offs needed for an SOA solution and a set of common categories (or types) of SOA services to help populate that architecture. ISO/IEC 18384-3 contains the SOA ontology, which is a formalism of the core concepts and terminology of SOA, with mappings to both UML (see Reference [16]) and OWL (see Reference [17]). ISO/IEC 18384-3 can be used independent of or in conjunction with ISO/IEC 18384-1 and this part of ISO/IEC 18384.

Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) —

Part 2: Reference Architecture for SOA Solutions

1 Scope

This part of ISO/IEC 18384 describes a Reference Architecture for SOA Solutions which applies to functional design, performance, development, deployment and management of SOA Solutions. This part of ISO/IEC 18384 includes a domain-independent framework, addressing functional requirements and non-functional requirements, as well as capabilities and best practices to support those requirements.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 18384-1, *Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) — Part 1: Terminology and concepts for SOA*

ISO/IEC 18384-3, *Information technology — Reference Architecture for Service Oriented Architecture (SOA) – Part 3: Service Oriented Architecture Ontology*

ISO/IEC 15474-1, *Information technology — CDIF framework — Part 1: Overview*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18384-1 apply.

3.2 Abbreviated terms

ABB	Architectural Building Block
B2B	Business To Business
BAM	Business Activity Monitoring
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CEP	Complex Event Processing
CICS	Customer Information Control System
CRM	Customer Relationship Management
EA	Enterprise Architecture
EAI	Enterprise Application Integration
EJB	Enterprise Java Beans
ERP	Enterprise Resource Planning

FCAPS	Fault, Configuration, Accounting, Performance, Security
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IT	Information Technology
ITIL	Information Technology Infrastructure Library
JAX-WS	Java Api For Xml Web Services
KPI	Key Performance Indicator
MDM	Master Data Management
NFR	Non Functional Requirement
POCO	Plain Old Clr Object
POJO	Plain Old Java Object
QOS	Quality Of Service
RA	Reference Architecture
RAS	Reliability Availability Scalability
SBB	Solution Building Block
SCA	Service Component Architecture
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Database Language SQL
WSDL	Web Services Description Language
WSRP	Web Services For Remote Portlet

4 Notations

Interpretation of diagrams should be done as follows.

4.1 UML

Most diagrams are not UML. Those that are have text to that effect before the diagram identifying the type of UML diagram so that the reader knows how to interpret it.

4.2 Entity-Relationship

Entity Relationship diagrams with boxes, lines arrows and circled numbers should be interpreted according to the following rules.

— Boxes are the metamodel concepts, layers, architectural building blocks, capabilities, or components.

— Arrows are relationships between metamodel concepts, where single arrow heads show direction of relationship; double-headed arrows indicate the relationship is bidirectional.

— Relationships are named, represented as labelled lines or arrows, and no cardinality is implied.

— Cardinality indications are participation in the relationship and well known mathematical conventions are used to express them (*== 0..*; 0..1==optional and only 1; 1==required as defined in ISO/IEC 15474-1).

4.3 Flows

Flows should be interpreted with the following rules:

— boxes that are layers, architectural building blocks or components;

— directional arrows showing the direction of the flow between the boxes;

— circled numbers on the flow arrows show the sequence of the flow and are used as a point of reference in any explanatory text.

4.4 Layer Diagrams

Layer diagrams, layered boxes and arrows, layer diagrams are usually some part of the SOA RA Layer and Aspects diagram in Figure 3 and should be interpreted using the following rules.

— Boxes that are layers or capabilities with architectural building blocks as smaller boxes positioned with in them. Horizontal boxes are functional layers. Vertical boxes or underlying boxes are cross-cutting aspects.

— Arrows between layers indicate interactions between the layers; capabilities of one layer are used by another.

— Arrows between ABB boxes within layers indicate interactions between ABBs within or across layers where single arrow heads show direction of interaction and double-headed arrows indicate the interaction is bidirectional.

4.5 Capability diagrams

Capability diagrams showing larger boxes as capabilities containing smaller boxes being architectural building blocks needed to fulfil that capability should be interpreted with the following rules.

— Relative positioning of the capability boxes is not relative.

— White indicates ABBs that are defined in the current layer. Some essential ABBs owned by other layers that are used to support the capabilities of this layer are shown in darker shades of grey. Additional ABBs defined in other layers may be used as needed.

— Grey ABBs are named using the name of the layer or aspect owning the ABB as a prefix followed by a colon and then the ABB name. For example: Governance: Registry/Repository indicates that the Registry/Repository ABB is owned by the Governance Aspect and the reader can go there to find out more information about it.

5 Conventions

The introduction is followed by a high-level summary of the 10 layers and the service types defined in this part of ISO/IEC 18384 for the convenience of the reader and to allow some readers who are only looking for a high-level 'executive level' understanding to easily read 7.1, 7.2, and 7.6. Each summary in 7.5 is repeated in the clause documenting the respective layers in the first clause labelled 'summary'.

This is followed by the definition and explanation of the metamodel used in this part of ISO/IEC 18384. The metamodel defines Layer, Capabilities and ABB concepts along with other core logical concepts. ABBs and capabilities are each defined uniquely in each layer. Capabilities and ABBs may require capabilities and ABBs defined in other layers in order to do fulfil their architectural requirements. The layers, capabilities and ABBs in this part of ISO/IEC 18384 are all logical elements and any reference to these logical element 'performing', 'supporting', 'interacting', or 'responsible for' means that when a SOA solution is developed, the physical realization of the capabilities and ABBs are actually 'performing', 'supporting', 'interacting', or 'responsible for'.

Each layer of the SOA RA is documented in a separate clause, Clause 5 through Clause 14. Each Layer is documented using the same organization:

1. Name of the Layer	
1.1 Overview	
1.1.1 Summary	
1.1.2 Context and Typical Flow	
1.1.3 Capabilities	- explains capabilities supported by layer
1.2 Details of ABBs and Supported Capabilities	
1.2.1 Details of ABBs	- detailed definitions of the ABBs
1.2.2 Structural Overview of the Layer	- overview of capabilities and ABBs they support
1.3 Inter-Relationships between the ABBs	- interactions between ABBs with in the layer
1.4 Significant Intersection Points with other Layers	
1.4.1 Interaction with Cross-Cutting Aspects	- interactions with Aspects supporting layer
1.4.2 Interaction with Horizontal Layers	- interactions with functional layers
1.5 Usage implications and Guidance	- best practices and advice
1.5.1 Options and Design Decisions	- considerations

At the beginning of each layer, there is small unlabelled figure of the SOA Solution stack for this SOA RA in the upper right hand corner that indicates with dark grey which layer in the reference architecture that this clause is defining.

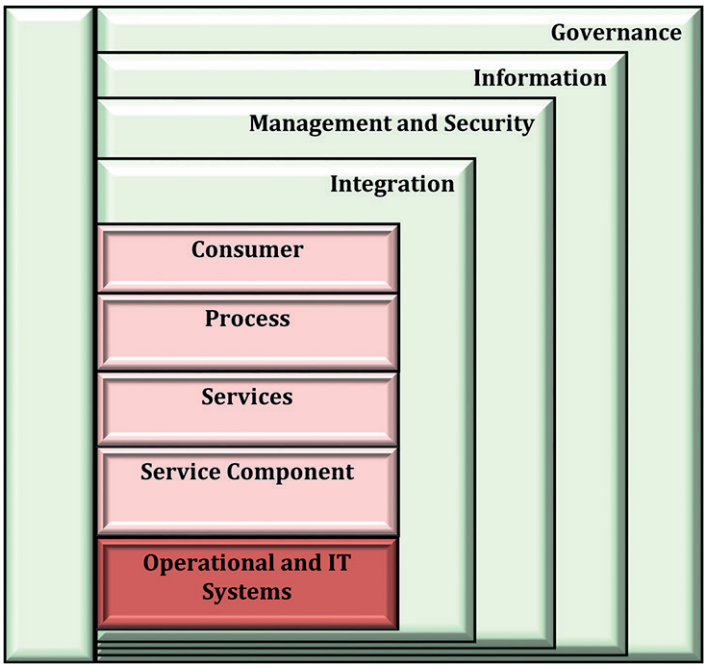


Figure 1 — SOA RA Layer/Aspect Indicator Diagram

For example, Clause 8 has Figure 1, at the beginning that indicates that the Clause is documenting the Operational and IT Systems Layer.

Clause 18 defines in detail the types of services commonly found in SOA solutions.

6 Conformance

ISO/IEC 18384 contains three parts which have different conformance requirements, as follows:

a) *Part 1: Terminology and concepts for SOA* – conformance only to terms and adherence to the semantics in the definitions;

b) *Part 2: Terminology and concepts for SOA* – conformance only to semantics of the metamodel and any Layers, ABBs, or capabilities that are used;

c) *Part 3: Service Oriented Architecture Ontology*, – conformance for OWL or non-OWL applications.

Conformance to this part of ISO/IEC 18384 is defined as follows.

This part of ISO/IEC 18384 is intended to be a set of best practices and guidance on creating successful architectures using the SOA paradigm. It is not intended to be mandatory or normative or to use for claiming conformance.

It is a qualitative standard in that users of the SOA RA may choose to deviate from the standard in certain areas. An organization may choose among the various ABBs provided by the SOA RA for conducting assessments, designing, or implementing architectures.

If a document, product or standard claims conformance with this part of ISO/IEC 18384 then it shall use the same semantics for the Metamodel, and any specific Layers, ABBs, or Capabilities used.

Some of the ABBs in the SOA RA are foundational and may be needed in most SOA solutions and others are only needed for some SOA solutions.

A service-oriented architecture or SOA solution does not conform to the partially layered architecture if there are certain things that are logically missing, then the missing building blocks that are key for the specific instance of the architecture should be identified. The corresponding Solution Building Blocks should be present.

Service-oriented architectures and SOA solutions can be different but conformant.

7 Overview

7.1 Introduction to SOA

Service Oriented Architecture (SOA) is an architectural style that supports service orientation and is a paradigm for business and IT (see ISO/IEC 18384-1:2016, 3.48). This architectural style is for designing systems in terms of services available at an interface and the outcomes of services. A service is a logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services and is a “black box” to consumers of the service. (see ISO/IEC 18384-1:2016 ,3.20).

In common with other architectural styles, SOA

— places unique requirements on system infrastructure,

— has environment-specific implementations, constrained or enabled by context and described within that context,

— requires appropriate governance of IT and systems and EA,

— has business solutions that are designed to mirror real-world business activities, and

— provides criteria to allow consumers to determine whether the business solution offered has been properly and completely executed in accordance with their expectations.

In addition, SOA has distinguishing characteristics that set it apart from other architectural styles, notably

— it promotes the use of open standards and interfaces in order to achieve interoperability and location opacity,

— services and processes are designed explicitly to operate both within or between organizations,

— it requires clear descriptions of the service offered,

— services and processes are designed to mirror real-world business activities,

— service representation uses business descriptions to provide context (i.e. business process, goal, rule, policy, service interface, and service component),

— it requires appropriate governance of service representation and implementation,

— service composition is used as a means to implement business processes, and

— it provides criteria to allow service consumers to determine whether the service has been properly and completely executed in accordance with the service description.

Service orientation is utilized for enabling efficient co-operation between autonomous (business) entities (e.g. clients, service providers, and third parties) that wish to collaborate to achieve common goals. Collaboration between the business entities can take the form of simple client-provider interaction, supply chains or virtual organizations that may take the form of bilateral or multi-lateral choreographies. In fact, organizations are a general concept and can represent multiple organizations or ecosystems of organizations.

Business-oriented SOA takes 'service' as its basic element to constitute and integrate information systems so that they are suitable for a wider variety of application requirements. Some of the benefits of using SOA are improvement in the efficiency of development of information systems, efficiency of integration and efficiency of re-use of IT resources. It also enables agile and rapid response of information systems to constantly changing business needs.

While an increasing number of solutions are being implemented using SOA in many different industries, a single set of SOA technical principles, specific norms, and standards have not been established for the world-wide market. Existing products and solutions have used various standards, methods and technologies. As a result, there is confusion about the effectiveness of SOA. To improve standardization and quality of solutions, it is necessary to establish a unified set of general technical principles for SOA, a standard reference architecture and a common set of service types based on best practices and experience.

It should be noted that the SOA principles defined in ISO/IEC 18384-1 are applicable to software engineering and can also be applicable to system engineering in order to formalize service-based systems (i.e. complex systems, federation of systems, systems of systems, enterprise architecture).

Service oriented computing is a software engineering paradigm for developing, delivering and governing services whose functionality is implemented as software components and where co-operation between business entities is enabled by information and communication technology. These activities can be private to an organization (e.g. deploying a service), collaborative between a set of business entities (e.g. service invocations and choreographies), or joint activities for maintaining the viability of the service ecosystem (e.g. publishing new services).

7.2 Introduction to the SOA Reference Architecture

The SOA Reference Architecture consists of two complementary parts: first, a reference architecture for SOA solutions and second, a set of common service types or categories. Both of these will support architects to develop architectures for SOA solutions and service-oriented business applications. This

includes all of the logical and physical design and runtime components, needed to develop, deploy, and operate service-oriented solutions for businesses. The complementary parts are enumerated as follows.

a) The SOA Reference Architecture enumerates the fundamental elements of an SOA solution or enterprise architecture standard for solutions and provides the architectural foundation for the solution by specifying the capabilities and architectural building blocks that support the realization of those capabilities.

   The SOA Reference Architecture (SOA RA) has 10 layers representing 10 key clusters of capabilities, considerations and responsibilities that typically emerge in the process of designing a SOA solution.

   Each layer supports, and is defined in terms of: requirements, logical, and physical concerns. The requirements concerns reflect what the layer enables and includes all of its capabilities. The logical concern includes all the architectural building blocks (ABBs), design decisions, options, and metrics. The physical concern of each layer includes the realization of each logical concerns using a particular chosen technology platform, standards and products that are determined by taking into consideration the different options and selecting one along with the documentation of architectural decisions that led to the selection. The actual realization of the architecture will be through a set of products or platforms that will be left open to the implementer of the standard.

   This Clause documents a high-level description of each of the layers. The detailed descriptions of the requirements and capabilities, logical architectural building blocks, and physical mapping are in Clauses 8 to 17.

b) Clause 18 documents a range of common domain specific (functional) and non-domain specific (supporting) categories of services that can be used as a checklist to help architects understand and decide which of these types of services are appropriate for their solution.

   Services are naturally a key concept in any service-oriented architecture and it is important to realize that there can be many different kinds. Services are categorized according to what they do, i.e. their function or purpose, in order to aid in ensuring both coverage and shared understanding. Of course, other categorization schemes are also possible and helpful.

   Partitioning services into groups and common categories is a common activity in the development of the services and service portfolio. These groups and common services type can help how both business and IT stakeholders have a common view and understanding of the architecture and the development and deployment of the portfolio of services that are part of that architecture.

   The layers and the service categories are intended to support the entire lifecycle of SOA solutions, from planning, design and development, though to deployment, execution, updating and eventual decommissioning.

   This SOA RA is derived from The Open Group SOA Reference Architecture technical standard (see Reference [8] for more information), China's Technical Reference Architecture for SOA Solutions (see ISO/IEC TR 30102:2012, Annex D; see Reference [30]), and SOA Related Function — Japanese Technical Reference Model (TRM) for the Government Procurement of Information Systems (see ISO/IEC TR 30102:2012, Annex F; see Reference [30]).

7.3 Metamodel

The metamodel defines the terms and relationships needed to understand this part of ISO/IEC 18384.

A textual description of the model with an illustrative diagram (UML) is provided. This includes a discussion on the capabilities concept as used in this part of ISO/IEC 18384. This subclause finishes with the architectural assumptions that the reader should understand before trying to understand the SOA RA.

Taking a pragmatic approach, for each layer, there are three concerns 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 concern will be left to the implementation of the standard by an adaptor of the standard).

The requirements concern reflects what the layer enables and includes all of its capabilities; the logical concern includes all the architectural building blocks, design decisions, options, KPIs, etc.; while the physical concern of each layer includes the realization of each logical concern 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 concerns of the SOA Reference Architecture and provides a model for including key architectural considerations and making architectural decisions through the elements of the metamodel. Examples of and best practices for these considerations and decisions are documented at the end of each layer.

Layers provide an abstraction and consolidation mechanism that consistently groups a set of related and cohesive capabilities and associates them with logical components called Architectural Building Blocks (ABBs). Thus, each layer provides a logical container for a unique set of ABBs and the capabilities they help realize. However, realizing some capabilities might require the use of ABBs assigned to multiple layers and cross-cutting aspects in order to realize those capabilities in their SOA solutions.

The separation of concerns[1)] and design considerations that entail from these layers are integral to an SOA. The SOA Reference Architecture has 10 layers, out of which three primarily support the provider side responsibilities of service interface and implementation and, three support service consumption and five are cross-cutting in that they provide support for all layers. The 10 layers comprise a comprehensive set of identified capabilities from which a SOA solution may be constructed. Most SOA architectures and solutions will use a subset of the ABBs in the layers. This may be due to the business context and constraints imposed on the solution. In addition, as an organization matures in their use of SOA and their SOA solutions become more sophisticated, they may choose to use a different set of ABBs.

Indeed, the need to combine the capabilities represented by ABBs in different layers demonstrates that a strict separation of concerns may not be possible for more complex (composed) capabilities.

It is important to realize that a Reference Architecture is not a Solution Architecture. Capabilities satisfy the requirements for functionality that these layers collect. Architectural Building Blocks (ABBs) are the logical components whose implementations carry out that functionality specified by the capabilities. An ABB can be realized by a Solution Building Blocks (SBB). An SBB is a physical or socio-technical construct whose structure and properties conform to the requirements associated with an ABB.

To summarize, a layer is an abstraction that is useful in grouping cohesive capabilities. Similarly, an ABB is an abstraction that is associated with a capability and the outcome the capability will produce. To be precise, it is the implementation of ABBs that realize the capability and its outcomes in SOA solutions. Layers, as a collection of ABBs, consolidate capabilities. The implementations constitute Solution Building Blocks (SBBs) that are "concrete", identifiable, and subject to governance and management.

When this part of ISO/IEC 18384 refers to a layer as "providing" or being "responsible" for a capability, it should be interpreted as actually the implementation of the necessary ABBs collected in the layer that is realizing the capability and its outcomes. Similarly, the ABBs, as abstract entities, represent elements that provide capabilities. When this part of ISO/IEC 18384 refers to ABBs providing, interacting, or being responsible for, in fact, it is the implementation of the ABBs that is providing, interacting, or behaving in a solution.

SOA solutions use ABBs in their abstract sense to communicate or document the design or architecture of the solution and use implementations of the ABBs to realize the solutions. Interactions between ABBs associated with a single layer are referred to as "interactions within a layer". However, solutions often make use of ABBs in multiple layers to represent more complex patterns; the use of such ABBs may be

1) If we are attempting to separate concern A from concern B, then we are seeking a design that provides that variations in A; do not induce or require a corresponding change in B (and usually, the converse). If we manage to successfully separate those concerns, then we say that they are "decoupled", for obvious reasons. As a simple example, a car's side view mirror controls are decoupled from its accelerator, permitting us to adjust our mirrors without fear that the car will speed up or slow down. [http://www.infoq.com/articles/separation-of-concerns]

referred to as “interactions across layers”. The simpler wording is often used for convenience; however, in practice, the interactions occur between ABB implementations.

To be explicit, the layers, as defined, are available to be used in any SOA solution. However, for any given problem/solution context, only a proper subset of all the capabilities of a layer may be required to be realized. For example, a simple SOA solution may not have any processes and therefore the capabilities and ABBs in associated with the Process Layer are logically not required or need to be realized.

Often, there is a need to use similar capabilities in multiple layers, like checking security access or accessing assets in a registry/repository. The ABBs that represent such widely used functionality are factored into special layers called cross-cutting aspects. For example, many layers might have local security permission checking and access to special registry/repositories but the capabilities and the associated ABBs are collected in the cross-cutting Management and Security Aspect.

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

This logical view of the SOA Reference Architecture addresses the question, “If I build an SOA solution, what would it look like and what abstractions should be present?” The question answered by this part of ISO/IEC 18384 is, informally, “If I assess an architecture proposing to be built upon SOA principles, what considerations and building blocks should be present and what should be assessed with?”

The SOA Reference Architecture has 10 layers representing 10 key clusters of considerations and responsibilities that typically emerge in the process of designing an SOA solution or defining 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.

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

Figure 2 uses UML (ISO/IEC 19505-2) to illustrate the relationship between the metamodel concepts.

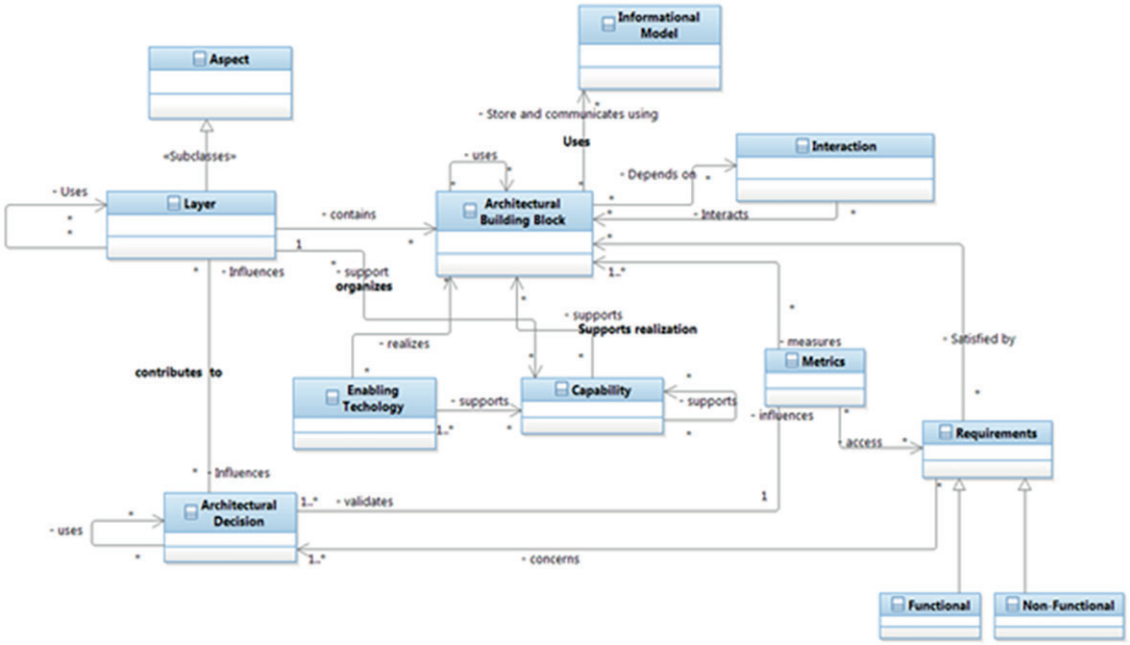


Figure 2 — Metamodel for instantiating the SOA Reference Architecture for a Given Solution

As shown in Figure 2, the metamodel of the SOA Reference Architecture includes the following elements.

— Layer: an abstraction of a grouping of a cohesive set of related capabilities and their identified ABBs, interactions among ABBs, interactions among layers, and the influences on and by architectural decisions.

— Aspect: is a layer that contains capabilities and functionality that are widely useful across functional layers and may need to be coordinated across multiple roles (see ISO/IEC 18384-1:2016 ,8.3).

— Capability: an ability that an organization, person, or system possesses to deliver a product or service. A capability satisfies a requirement or category of requirements that fulfil a strongly cohesive set of needs.

— Architectural building block (ABB): a logical element that supports realization of one or more capabilities. Implementations of ABBs provide the basic building blocks for composing SOA solutions. 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 can be described not only in terms of static relationships between ABBs but also in terms of the interaction sequences between the ABBs that address specific scenarios. 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.

  NOTE ABBs from other layers are named with an owning layer name prefix, e.g. Governance Aspect: Registry/Repository.

— Solution Building Block (SBB): Solution Building Block realizes one or more architectural building blocks by providing a technical, organizational, or social implementation, a system, that conforms with the requirements associated with the ABB through the capabilities it supports. A solution building block can be implemented with enabling technology, such as a service directory, for example. (See ISO/IEC/IEEE 42010.)

— Architectural decision: a decision derived from the identification and analysis of the available options. The architectural decision is driven by architectural requirements, and involves governance

rules and standards, ABBs, Key Performance Indicators (KPI), and Non Functional Requirements (NFRs) 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: an abstraction of the various relationships between ABBs. This may be represented by the use of diagrams, other patterns, pattern languages and interaction protocols.

— Metrics: analytical measurements intended to quantify the state of a system and are often called Key Performance Indicators (KPI). Architectural decision will help decide what key performance indicators should be defined and over time the KPIs will validate that the right decisions were made and if they should be adjusted.

— Requirements: documented needs that a particular layer, ABB, or SOA solution satisfies. Requirements may act as input to an architectural decision.

— Functional requirements: requirements that specify functions that systems or system components perform (see ISO/IEC 25000).

— Non-functional requirements (NFR): specify the overall quality requirements of the SOA solution or system. NFRs help address architectural cross-cutting concerns such as security and management.

— Enabling technology: technology used to realize ABBs.

— Information Model: a structural model of the information associated with ABBs including data exchange between layers and external services. The information model includes the metadata about the data being exchanged.

7.4 Capabilities

A capability is an ability that an organization, person, or system possesses. Capabilities are typically expressed in general and high-level terms and may require a combination of organization, people, processes, and technology to achieve. There are pure business capabilities such as Process Claim or Provision Service Request and there are technical capabilities such as Service Mediation or policy based Content Based Routing. Both business and technical capabilities are represented in SOA and enabled and supported by SOA.

Using a capability modeling as part of the approach has some major advantages.

— Allows architects to focus on the “what” rather than the “how”. This supports an abstract approach that is focused on the requirements of the solution.

— Enables business capabilities to be aligned to the technical capabilities required to service them.

— Enables architects to derive and re-balance the plan for SOA adoption in an agile fashion. For example, if an organization foresees the need for integrating services across different business units, it might require a certain set of SOA layers and architectural building blocks to be enabled.

This SOA Reference Architecture enables deriving the solution architecture using capabilities; however, the capability mapping process itself is out of scope for this part of ISO/IEC 18384 and is normally a part of the organizational service modeling methodologies. For example, the business capability to cross-sell requires a technical capability to have a common shareable set of data, where the data is from different systems in an enterprise. This, in turn, requires shareable metadata about data, supporting “information services” in some form and the ability to transport, mediate and share data from the disparate systems in a common form. Thus, a business capability (cross-selling) is dependent on technical capabilities [the need to be able to have a common view of data (information services), the need to mediate, integrate and transport the data, etc.]. Each of these capabilities maps to architectural building blocks supported by the layers of the SOA RA.

A capability-based approach enables architects to determine when they need a particular SOA RA layer and functions and to help facilitate making decisions when organizational priorities change. In this part of ISO/IEC 18384, the SOA RA enables architects to determine if there are interdependencies and technical requirements for a layer and its constituent building blocks, beyond those defined by business capabilities, to create a coherent set of capabilities which the SOA RA needs to satisfy.

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. 7.5 explores the details of these layers and their constituent elements.

7.5 Reference Architecture for SOA Solutions

7.5.1 Overview of Reference Architecture

This subclause presents an overview of the reference architecture for SOA solutions. The reference architecture is a high level abstraction of an SOA, partitioned and factored into layers. Each layer provides a set of required capabilities that addresses a specific subset of characteristics and responsibilities.

The following subclauses refer to Figure 3 and provide detailed descriptions for each of the 10 layers and their interactions.

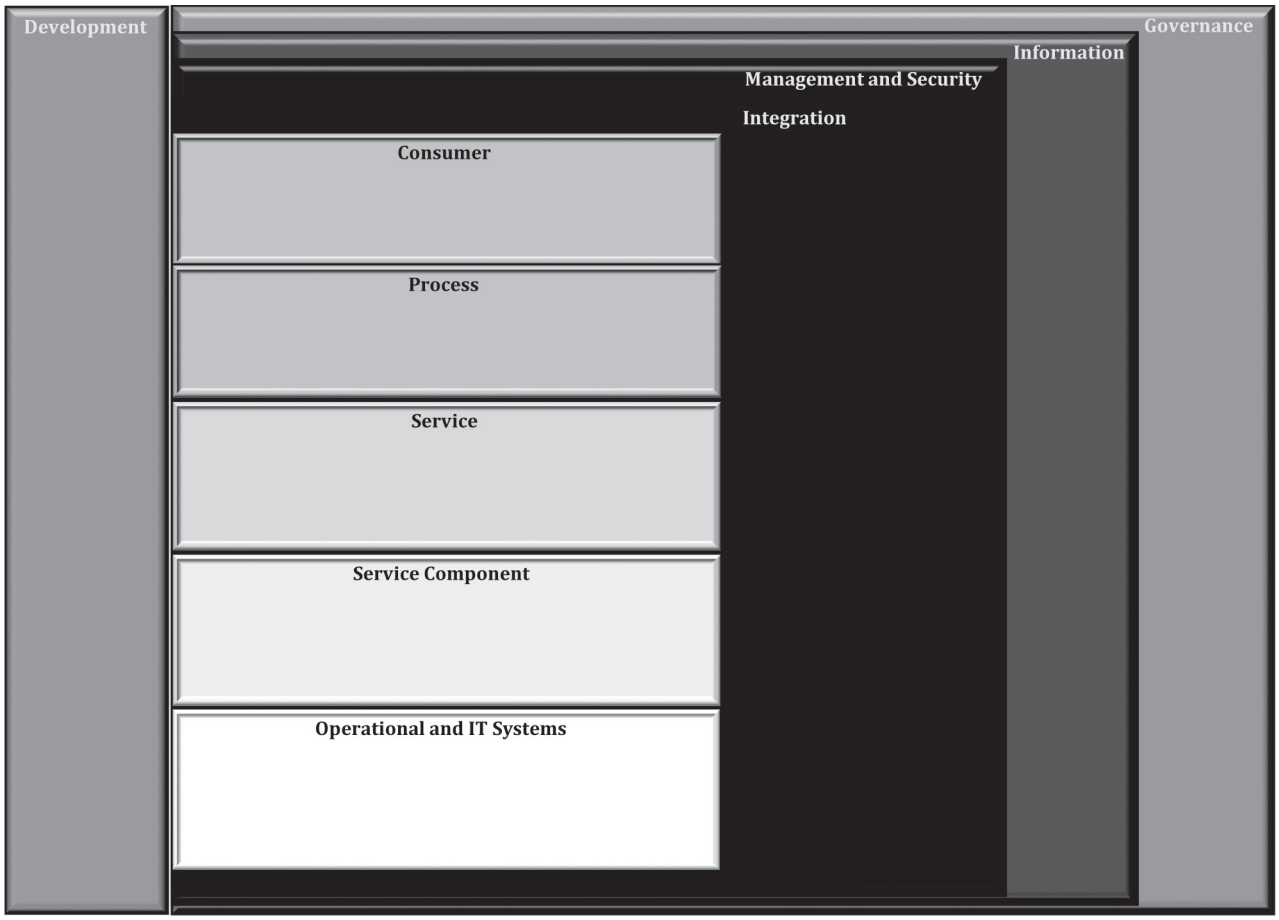


Figure 3 — Reference Architecture for SOA Solutions

Figure 3 depicts an SOA Reference Architecture for SOA solutions as a set of logical layers as a partially-layered architecture. This means that one layer does not solely depend upon the layer below it. Therefore, a consumer can access both the Process Layer and the Service Layer directly. This is important because a given SOA solution may not need the capabilities and ABBs in the Process Layer and would not benefit

from the business value proposition associated with the Process Layer. The Consumer Layer may then interact directly with the Service Layer for efficiency and architectural simplicity. Capabilities and ABBs of the Process Layer can be added to the architecture at any time in the future when there is a need for the business value they provide. 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.

Three of the horizontal layers (dark grey) address the implementation and interface with a service (the Operational and IT Systems Layer, the Service Component Layer and the Services Layer). Three of the horizontal layers support the consumption of services (the Service Layer, Process Layer, and the Consumer Layer). Service layer is used by both. Five of them (light grey) support cross-cutting Aspects of a more supporting (sometimes called non-functional or supplemental) nature (the Information Aspect, the Management and Security Aspect, the Integration Aspect and the Governance Aspect). One vertical layer supports development for both functional layers and Aspects. 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.

The SOA Reference Architecture does not assume that the provider and the consumer are in one organization (or many organizations) and supports both SOA within the enterprise, as well as across multiple enterprises in the industry ecosystem. The need for both intra-and inter-enterprise SOA is important, since SOA a foundation of Cloud Computing as being defined in ISO/IEC 17789 (see Reference [29]). Decoupling providers from consumers using services is essential when the providers and consumers are in different organization. Decoupling is just as important for providers and consumers in the same organization when they need separation along the lines of business relationships. These organizations use this architectural style (where one is the consumer and the other is the provider) and customize it for their own needs for integrating and interacting among themselves. The lower layers (Services Layer, Service Component Layer and Operational and IT Systems Layer) are concerns for the provider and the upper ones (Service Layer, Process Layer and Consumer Layer) are Aspects for the consumer. Note that the Service Layer is a shared layer of concern between both Providers and Consumers. Each layer and the relationships between the layers is described in the subsequent clauses.

Note that there are five horizontal layers that are functional in nature and relate to the functionality and the overall capabilities that are provided and implemented by the SOA solution. The underlying layers support the horizontal layers by providing a set of cross-cutting Aspects. These Aspects span and apply to all of the horizontal functional layers but are clustered around independent notions such as information, integration and governance in their own right and serve as cross-cutting Aspects of the SOA architectural style. The layering of these cross-cutting aspects indicates that each aspect on the bottom applies to all of the aspects above it. For example, governance applies to and should be considered for all elements in all of the layers above it. Information is important for Management and Security, Integration and the functional layers. Management and Security apply to integration and the functional layers. However, each layer can use elements in the layers above and beneath them, so Governance uses metrics monitoring in the management layer and the management layer uses policies defined in the Governance aspect.

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

7.5.2 Operational and IT Systems Layer

The Operational and IT Systems Layer captures the new and existing organization infrastructure needed to support the SOA solution at design, deploy and run time. This includes the following:

— all infrastructures to run the SOA and its components;

— all operational and runtime hosting of components, both physical and infrastructural;

— all deployment time components;

— all resources to support services, data, and application systems;

— all assets required to support the functionality of the service in the SOA, where the assets may include custom or packaged application assets, new services, services created through composition or orchestration, infrastructure services, etc.

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 Aspects. 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.

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

— existing monolithic custom applications;

— existing transaction processing systems;

— existing databases;

— existing package applications and solutions including ERP and CRM packages;

— legacy applications and systems;

— access to existing web services;

— systems built from web services;

— IT Infrastructure;

— Enterprise Application Integration (EAI) systems;

— Service Resources providing access to existing business elements or third-party elements;

— Data Resources providing physical storage of data in the business solution;

— Application System Resources providing specific business functions;

— access to existing business services.

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 construct and the rest of the SOA in the wider context of cloud computing. Key requirements for this layer are outlined in the capability clause describing the capabilities provided to fulfil those requirements.

7.5.3 Service Component Layer

The Service Component Layer contains capabilities that support components which represents the implementation or “realization” for services or operation on services; hence, the name Service Component. The layer 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 management and quality of service interactions. They "bind" the service interface to the implementation of the service in the Operational and IT Systems Layer. Service components may be hosted in containers which support a service description (see 10.2.2.1 for more information on containers).

The Service Component Layer manifests the IT conformance with each service interface defined in the services layer; it helps guarantee the alignment of IT implementation with service description.

Each Service Component

— realizes one or more services,

— provides an enforcement point for "faithful" service realization (ensures quality of service and service level agreements),

— enables IT flexibility by strengthening the decoupling in the system by hiding volatile implementation details from service consumers,

— offers a façade behind which technologies can be deployed as required to enable service functionality, and

— generally contains business-specific logic with no reference to integration logic.

The Service Component Layer enables flexibility through encapsulation and by enabling loose coupling. A separation of concerns is achieved such that the service consumer can assume that the realization of the service is faithful to its published description (service compliance) and the service provider will ensure that such compliance is achieved. The details of the service realization are of no consequence to the consumer. The service provider is consequently able to replace one component with another having the same interface, producing the same outcomes (real world effects), and having identical conditions of use without any impact on service consumers.

7.5.4 Services Layer

The Services Layer consists of the logical representation for all the services. The Service Layer can be thought of as containing the service descriptions for business capabilities, services and IT manifestation used and created during design time, as well as runtime service contracts and descriptions that are used at runtime.

The Services Layer is one of the horizontal layers which provide the business functionality supported in the SOA and describes functional capabilities of the services in the SOA.

The description provides consumers with information needed to invoke the business functions exposed by a provider of the service; ideally, this may be done in a platform independent manner. Service descriptions may include or have links to the following:

— descriptions of the abstract functionality offered by the service similar to the abstract stage of a WSDL description (see Reference [14]). Note that the use of WSDL is illustrative and the description can be done in any language supporting description of the functionality;

— policy documents;

— SOA management descriptions;

— attachments that categorize or show service dependencies.

Some of the services in the Service Layer may be versions of other services in the set implying that a significant successor/predecessor relationship exists between them.

This layer contains the contracts which include service descriptions that bind the provider and consumer. Services are offered by service providers and are consumed by service consumers (service requestors). Service Components or existing enterprise applications (such as legacy systems and packaged

applications) are responsible for the actual implementation or realization of a service. The Operational and IT Systems Layer supports the runtime environment; therefore, implementation of the service components may reside in or use a container and other ABBs in the Operational and IT Systems Layer.

The Services Layer supports the following:

— functional capabilities or services that enable business capabilities those businesses perform in order to achieve a business outcome;

— supporting capabilities to define and specify the “services” in terms of service description;

— supporting capabilities to enable the runtime execution of service and the support of service virtualization.

7.5.5 Process Layer

The Process Layer covers the process representation, composition methods, and building blocks for aggregating loosely coupled services as a sequence of steps aligned with business goals. Data flow and control flow are used to enable interactions between any combination of services and business processes. The interaction may exist within an enterprise or across multiple enterprises.

Business capabilities are realized through the execution of one or more business processes. These business processes may be realized through service compositions (e.g. orchestrations, choreographies, or collaborations) and include support for manual human interactions and long-lived transactions. The evolution of service composition into flows (e.g. choreographies of services bundled into a flow) can act together to establish an SOA solution. These SOA solutions support specific use cases and business processes.

This layer includes information exchange flows between participants (individual users and business entities), resources, and processes where the exchanged information may include unstructured and non-transactional messages. The business logic is used to form service flows as parallel tasks or sequential tasks based on business rules, policies, and other business requirements.

The Process Layer performs three-dimensional process-level handling: top-down, bottom-up, and horizontal. From the top-down direction, this layer provides capabilities and ABBs to help 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 service-oriented collaboration control between business processes, services, and service components.

In summary, the Process Layer in the SOA Reference Architecture plays a central coordinating role in connecting business level requirements and IT-level solution components through collaboration with the Integration Aspect, Management and Security Aspect, Information Aspect, Governance Aspect, and Services Layer.

7.5.6 Consumer Layer

The Consumer Layer is the point where consumers, either human actors or SOA solutions, interact with the SOA solution ecosystem. It enables SOA solutions 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). In this discussion, channels can be thought of as the platforms by which SOA consumers access services through the SOA. Examples of channels include front-ends and interactive voice response (IVR) systems, which could both leverage the same core functionality within the SOA. Thus, the Consumer Layer is the point of entry for all internal and external interactive consumers, including services acting as consumer (for example, in B2B scenarios).

For human consumers, the Consumer Layer is often realized through a user interface that accepts requests and returns responses. This interface may enable specific users to customize preferences, integrate with consumer channels, including rich clients such as mashups and Ajax (see Reference [18])

and act as a mechanism for the underlying SOA to expose its functionality. Standards such as Web Services Remote Portlets (WSRP) (see Reference [15]) may leverage services at the application interface or presentation level.

The user interface provides the visible portion of the Consumer Layer capabilities but the Consumer Layer may also incorporate other business processes dictated by policy or desired business outcomes. For example, consumer layer capabilities may include the point where in-bound service 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 via collaboration with other Aspects.

For consumers that are other services or SOA solutions, the Consumer Layer points to defined service interfaces, where in parallel with the human use of the Consumer Layer, the service interface may point to a composition which includes the application of other business processes, such as security and quality of service from the Management and Security Aspect. The Consumer Layer provides the capability to quickly create a front end for business processes and other service compositions to respond to changes in business requirements. This "front end" can be a new service interface, a new user interface, or an appropriate combination. It enables channel independent access to those business processes supported by a variety of applications and platforms.

This decoupling between the consumer and the rest of the underlying SOA provides organizations the ability to support agility, enhance reuse and improve quality and consistency.

7.5.7 Integration Aspect

That Integration Aspect enables the loose coupling between the request and the concrete provider by matching the Service Request and Service Implementation. This loose coupling provided by the Integration Aspect is not only a technical loose coupling addressing protocols, binding, locations or platforms, but can also be a business semantic loose coupling performing required adaptations between service requester and provider.

There are numerous sets of capabilities the Integration Layer supports to overcome structural and semantic mismatches at service interfaces. For example, the Integration Aspect supports the integration with solution platforms by the other layers in the SOA RA using mediators, transformers, or adapters to enable the access of services by other ABBs, layers and the capabilities associated with service transport. Mediation includes transformation, routing, and protocol conversion. Integration includes adapters and service enablement. Routing includes service interaction and service virtualization. Transport includes service messaging and message processing. It can be thought of as the plumbing that interconnects the SOA solution. Both of these can be driven by policy.

Integration support includes capabilities that enables and provides the capability to mediate between the service requester and the service provider. It provides capabilities to transform, route, and convert protocols, enabling support for heterogeneous environment, adapters, service interaction, service enablement, service virtualization, service messaging, message processing and transformation.

Routing support includes supporting capabilities through which a consumer/requestor can connect to the correct service provider. These can start with point-to-point capabilities for tightly coupled endpoint integration and cover the spectrum to a set of intelligent routing, mediation, and other transformation mechanisms often associated with, but not limited to, mediation services provided by an Enterprise Service Bus (ESB). The service description specifies a location where a service is provided and an associated binding and for part of a service contract. A mediation service, on the other hand, provides a location independent mechanism for integration, service substitution, and virtualization.

Transportation support that occurs here is primarily the integration of service component, service, and Process Layers (the "functional" layers). 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, CRM client (used by Customer Service Rep), etc. The transformation of response to HTML (for web), Voice XML (for IVR), or XML string can be done via XSLT functionality supported through mediation service transformation capability in the Integration Aspect.

7.5.8 Management and Security Aspect

The Management and Security Aspect supports non-functional requirement (NFR) related issues as a primary concern of SOA and provides a focal point for dealing with them in any given solution. It contains the capabilities for ensuring that a SOA meets its requirements with respect to monitoring, reliability, availability, manageability, transactionality, maintainability, scalability, security, safety, life cycle, auditing and logging, etc. It includes the same scope as the traditional Fault, Configuration, Accounting, Performance, and Security (FCAPS) from ITIL (see Reference [27]) or Reliability, Availability, Serviceability (RAS) (see Reference [28]).

Management and Security is especially important for SOA solutions to enable the loosely coupled solutions to scale and efficiently fulfil non-functional requirements. This layer provides capabilities that maintain and ensure the "quality of service (QOS)". In order to enable both management and security, this layer provides capabilities that

— provides solution management of various concerns, such as availability, reliability, security, and safety, as well as the mechanisms to support, track and monitor, and manage solution quality controls,

— provides the ability to monitor and enforce a multitude of policies and corresponding business rules including business level policies, security policies, access privileges and data access policies,

— serves as an observer of the other layers and can create events when a non-compliance condition is detected or (preferably) when a non-compliance condition is anticipated,

— provides the service and SOA solution lifecycle processes with the capabilities required to ensure that the defined policies, non-functional requirements (NFRs), and governance regimens are adhered to for service and SOA solution lifecycle processes,

— supports the ability to monitor and manage at both the business level [in terms of key performance indicators (KPIs), events and activities in business processes] and the IT systems level (for the security, health and well-being of IT systems, services, applications, networks, storage and processors), and

— supports monitoring and capturing of service and solution metrics in an operational sense and signalling of non-compliance with non-functional requirements relating to 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.

Finally, the same kinds of management and monitoring that apply to businesses are important for managing services and SOA solutions and may need extensions to handle the service-oriented nature and the cross domain boundaries of many SOA solutions. These traditional capabilities supported by SOA solutions include

— IT Systems Monitoring and Management,

— Service and SOA Solution Monitoring and Management,

— Business Activity Monitoring and Management,

— Event Management,

— Configuration and Change Management,

— Policy Monitoring and Enforcement,

— Lifecycle management, and

— Auditing and Logging.

SOA security addresses the protection of the SOA solution against threats across the vulnerability dimensions of a service-oriented architecture. This includes protecting the interactions between service consumers and service providers, as well as protecting all of the elements that contribute to the architecture. Examples of threats to be protected from are destruction, corruption, removal, disclosure, and interruption. Some of the security dimensions to help protect against these threats include access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy.

The capabilities that explicitly address security are as follows.

— Security Management: Manages and monitors security and secure solutions. This provides ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, enable the IT organization to manage IT-related risks and compliance, and provide the automation and audit basis for security management.

— Facilities Security Management**:** This category of capabilities provides the command centre for security management, as well as the operational security capabilities for non-IT assets and services to ensure protection, response, continuity and recovery. It also supports for security of physical assets such as locations, facilities, services, inventory, physical access control, human identity, etc.

Important areas for policy enforcement are security, auditing, messaging transportation, infrastructure availability, service availability and reliability. Responses (dispensations and appeals) to non-compliance and exceptions are also defined by the governance aspect.

7.5.9 Information Aspect

The Information Aspect provides capabilities for enabling development of a unified representation of the information assets of an organization as represented by its IT services, systems and SOA solutions. The unified representation for an organization may require rationalization and ongoing coordination of assets from across many organizations. Representing information enables business needs and processes to be in alignment with one or more business vocabularies.

The Information Aspect includes information architecture, business analytics and intelligence, metadata considerations. It focuses on the inclusion of key considerations pertaining to information architectures that can be used as the basis for the creation of business analytics and business intelligence through the analysis of data held in repositories. These repositories include metadata content that is stored using capabilities provided in this layer. The Information Aspect supports the establishment of information services capability architectures that can be used as the basis for the creation of business analytics and business intelligence through data marts and data warehouses. 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 a systematic improvement in data quality.

The Information Aspect supports the following capabilities:

— ability to support information services capability that will support a shared, common and consistent expression of data;

— ability to integrate information across diverse actors and organizations in order to effectively communicate across the different organizational domains;

— ability to define metadata that is used across the SOA RA and, in particular, the metadata that is shared across the layers;

— ability to enable secured and protected information via interaction with the Management and Security Aspect;

— ability to support business activity monitoring and critical to the usage of the RA and its realization.

An information virtualization and information service capability may involve the ability to retrieve data from different sources, transform it into a common format and expose it to consumers using different protocols and formats.

7.5.10 Governance Aspect

SOA Governance defines policies, guidelines, standards and processes that reflect the objectives, strategies and regulations to which services and SOA solutions align with business needs over time. The success of a SOA solution is often in terms of meeting business value objectives. SOA governance activities should conform to Corporate, IT and Enterprise Architecture governance principles and standards relevant to the SOA ecosystem in which the services and SOA solutions are intended to interact. The SOA Governance should also be adapted to match and support an appropriate SOA maturity level.

The Governance Aspect includes both SOA solution governance (governance of processes for policy definition and enforcement), as well as service governance (service life-cycle). This covers the entire lifecycle and portfolio management of the service and SOA solutions (e.g. SLAs, capacity, performance, security and monitoring). This Aspect also supports governance that needs to be coordinated across organizations, where service consumers and service providers are using services from other organizations.

The goal of the SOA Governance Aspect is to ensure consistency of the service and solution portfolio and supporting lifecycle processes. A given service or SOA solution may be part of more than one portfolio but it needs to conform to the defined governance for each Corporate or Enterprise domain to which it applies. Thus, some services or SOA solutions may not be appropriate for every governance domain.

The SOA Governance Aspect provides an extensible and flexible SOA governance framework based on ISO/IEC 17998 that supports the alignment of business and IT, including the following:

— Service Level Agreements based on requirements for quality of service and key performance indicators (KPIs);

— capacity and performance management policies;

— design time concerns such as business rules.

As a part of the governance framework, a governance regimen (i.e. the customized compliance, dispensation and communication processes to govern the SOA lifecycle and portfolio management) will need to make use of capabilities to store and access governance artefacts, as well as capabilities for managing and enforcing policy, monitoring metrics and managing the configuration and governance of the solution. Organizations may also need a strong change control capability to support changes to governance and the ensuing management of those changes.

To ensure ongoing business and IT alignment, the governance processes and business condition should be continuously evaluated and updated. These separate processes may use the same capabilities as the governance regimen and Management and Security Aspect.

The Governance Aspect supports the following capabilities:

— definition of policies, compliance and exceptions characteristics;

— monitoring the health of SOA services, solution and governance via the Management and Security Aspect;

— identification of metrics reporting on compliance, exceptions, service health, and versions;

— an incorporation of for business rules into the governance structure.

7.5.11 Development Aspect

The Development Aspect contains all of the components and products needed to develop and change implementations of SOA services and solutions. The service implementations should include the development of or use of implementations in the Operational IT and Systems layer, Service Component Layer, Service Layer, Process Layer and cross cutting aspects. Service implementations should

encapsulate existing systems and resources such that late binding of services can be supported to promote loose coupling.

Development includes solution and service design, modeling, implementation and deployment. Operational and management capabilities are the responsibility of the Management and Security Aspect. Maintenance uses capabilities from Development Aspect and Management and Security Aspect.

Tools that support the Development Aspect include the entire suite of architecture tools, modeling tools, development tools, visual composition tools, assembly tools, methodologies, debugging aids, instrumentation tools, asset repositories, source code control, discovery agents, and publishing mechanisms that may be used to construct a SOA solution.

The Development Aspect supports the following capabilities:

— development, configuration, debugging and testing environments for the construction of services;

— testing of services and SOA solutions, ranging from isolated testing to testing within the operational environment or ecosystem;

— coordination with monitoring to effectively provide for continuous testing throughout the operational lifetime;

— service encapsulation of existing application systems or data resources;

— reusing existing assets to develop services.

7.6 Common Services Categories

Service is naturally a key concept in any service-oriented architecture and it is important to realize that services can be categorized in many different ways. Clause 18 defines a standard categorization scheme for services where they are categorized according to what they do, i.e. their function or purpose, in order to aid in ensuring both coverage and shared understanding. This Clause provides a brief listing of the functional categories; for details, see Clause 18.

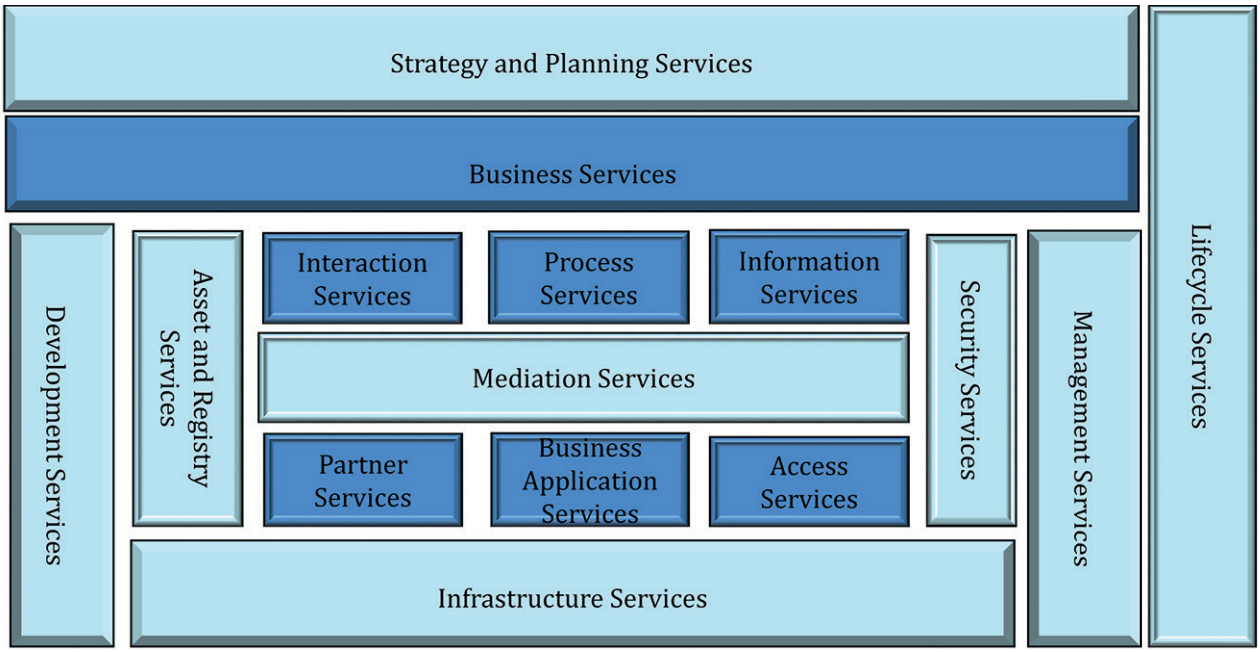


Figure 4 — Common Service Categories

The categories of services are broken down in Figure 4. Dark coloured services are considered to be domain specific where implementations of the services are required to be unique to the domain or solution being developed.

The remaining services categories are considered to be domain independent where implementations of services of this category can be used directly in many different domains or solutions and may be used without extensive customization.

Services categories are as follows.

— Mediation Services - category of services that provide the functions related to connecting service consumers with service providers where connecting can support effective and optimized mediation, transformation and routing of requests across the network and meet the goals of the business.

— Interaction Services - category of services that provide the presentation logic of the business design, supporting the interaction between other solutions and end-users.

— Process Services - category of services that execute various forms of compositional logic, such as business process flows.

— Information services - category of services that contain the data logic of business solution including the provision of access to the persistent data of the business, the support of data composition and to provide an architecture for managing the flow of data across an organization.

— Access services - category of services that encapsulate adapters to integrate legacy and new functionality into the SOA solution, including by wrapping or augmenting the logic of the existing functions to better meet the needs of the business design.

— Security services - category of services that address the protection against threats across the vulnerable dimensions of an SOA, including interactions between service consumers and service providers, as well as all of the elements that contribute to the architecture.

— Partner services - category of services that enable custom interaction between business partners, such as the semantics of partner interoperability that have a direct representation in the business design.

— Lifecycle services - category of services that support managing the lifecycle of SOA solutions and all of the elements that comprise them across development and management ranging from strategy to infrastructure.

— Asset and registry/repository services - category of services that manage and provide access to the information assets stored in configuration management databases, registries and repositories.

— Infrastructure services - category of services that form the core of the information technology environment for hosting SOA solutions.

— Management services - category of services that represent the set of management tools used to monitor metrics, service flows, the health of the underlying system, the attainment of service goals, the enforcement of administrative policies and recovery from failures.

— Development services - category of services that encompass the entire suite of architecture tools, modeling tools, development tools, visual composition tools, assembly tools, methodologies, testing, debugging aids, instrumentation tools, and discovery agents needed to construct services and SOA solutions.

— Strategy and planning services - category of services that support creating vision, blueprint and transition plan for improving business outcomes.

— Business application services - category of services that implement core business logic where these service implementations are created specifically within a business model.

— Business services - category of services that capture the business function and are offered to external consumers, often referred to as higher level or coarse-grained services.

Note that all common services should provide well-defined functionality that can be customized for consumer needs rather than a new service for every variation. For example, an Interaction Service can access data in XML and apply custom style sheets to give the desired presentation; it does not require a different service for each presentation.

See Clause 18 for more information on these service categories and how they relate to the underlying implementations and architectural building blocks of SOA.

7.7 Assumptions and Key Concepts

7.7.1 General

This Clause discusses the consequences of using the major metamodel concepts of "Capability" and "Requirements" on this part of ISO/IEC 18384 and the approach taken to describe the layers of the SOA Reference Architecture and how those layers may interact.

7.7.2 Functional and Non-functional

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

Security refers to the ability to ensure that a service can be accessed in a secure way (including authorization and authentication) and that the information provided by a service is only available by those that should be able to access the data. Availability refers to the percentage of how a specific service can be available during a specific time frame. Reliability refers to the ability of a specific service that stays no fail during a specific time frame. Scalability refers to the ability of a specific service that supports various scales of consumer groups. Latency refers to the delay of accessing a specific service due to internal implementation details and processes. Governance is defined in 7.5.10 and Clause 16.

7.7.3 Requirements

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

— a set of service requirements which includes business (a.k.a functional) and NFRs on a service;

— service requirements result in the documented capability that a service needs to deliver or is expected to deliver;

— 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;

— 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 combination of capabilities from one or more layers in the SOA Reference Architecture (SOA RA).

7.7.4 Services

Services themselves have a description element and a functional element. The service description states what the service does for consumers while the functional element implements what a service is obligated to provide based on the service description. The service description identifies a service

interface and associated binding through which the underlying functional element realizes the capability. This model addresses services exposing capabilities implemented through legacy assets, new assets, services composed from other services, or infrastructure services.

7.7.5 Documenting the Layers

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

Thus, in the clauses of this part of ISO/IEC 18384 that describe the layers in detail, we

— provide an overview and description of the layer and motivation behind the layer,

— describe the key capabilities supported by the layer,

— provide a structural overview of the layer which includes detailed descriptions of ABBs that enable the capabilities of the layer, and

— describe the interactions within the layer and across layers of the SOA RA.

In general, a theme is followed where each layer has a part which supports a set of capabilities/ABBs which support the interaction of the layer with other elements in the SOA RA, a part which supports the actual capabilities that the layer satisfies, and a part which supports the orchestration and management of the other ABBs to support the layer's dynamic, runtime existence. Thus, in the clauses of this part of ISO/IEC 18384 that describe the layers in greater detail, we

— provide an overview and description of the layer and motivation behind the layer,

— provide the key capabilities supported by the layer,

— provide a structural overview of the layer which includes detailed description of ABBs enabling the responsibilities of the layer, and

— describe the interactions within the layer and across other layers in the SOA RA.

7.7.6 Logical and Physical Elements

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

— All runtime elements that are part of the physical or operational or deployment architecture are actually represented in the Operational and IT Systems Layer of the SOA RA.

— The SOA RA will provide a logical abstraction of the runtime at a moment in time and expand that view in terms of a set of layers that are depicted by horizontal/functional layers and a set of supporting or cross-cutting layers (backdrop) of the SOA RA. The five horizontal layers deal with enabling the business capabilities required by the application running on the architecture. The five supporting layers support the functionality that is provided by the horizontal layers. The horizontal or functional layers include the Service Component Layer, the Services Layer, the Process Layer, and the Consumer Layer.

— Each part of the runtime is abstracted into a layer whose capabilities differ significantly from that of the other layers. For example, the Consumer Layer provides capabilities that support interaction with consumers of the service, whereas the Service Component Layer provides capabilities that support the implementation of the service in a Service Component. That Service Component, in turn, will be running in a container within the Operational and IT Systems Layer, at runtime.

7.7.7 Interactions between Layers

From 7.3, layers are an abstraction useful in grouping cohesive capabilities. Layers, as a collection of ABBs, consolidate capabilities. The implementations of ABBs are Solution Building Blocks (SBBs) that are "concrete", and identifiable, and subject to governance and management. When this part of ISO/IEC 18384 refers to a layer as "providing" or being "responsible" for a capability, it should be interpreted as "the implementation of the necessary ABBs collected in the layer that is realizing the capability and its outcomes".

SOA solutions use ABBs in their abstract sense to communicate or document the design or architecture of the solution and use implementations of the ABBs to realize the solutions. Interactions between ABBs associated with a single layer are referred to as "interactions within a layer". However, solutions often make use of ABBs in multiple layers to represent more complex patterns; the use of such ABBs may be referred to as "interactions across layers". The simpler wording is often used for convenience; however, in practice, the interactions occur between ABB implementations.

For example, Figure 5 describes an interaction between the Consumer Layer and the Process Layer using the Integration Aspect.

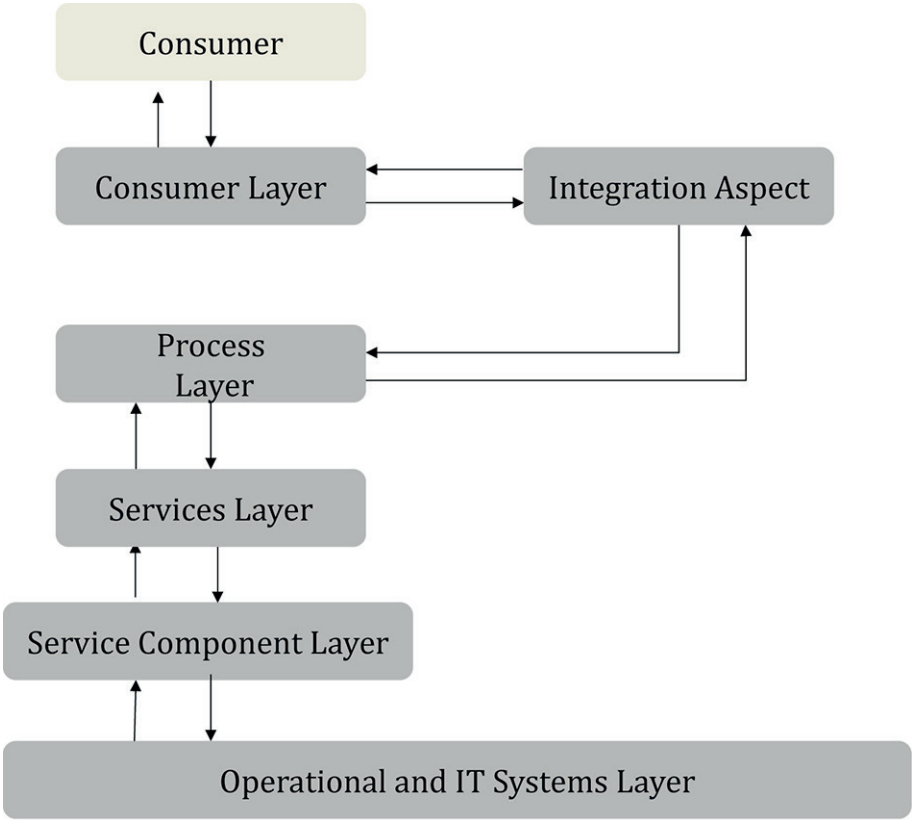


Figure 5 — Typical Interactions among the Layers of the SOA RA

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

— Service consumers request services using the Integration Aspect.

— The Integration Aspect invokes the business process in the Process Layer which is using one or more services.

— Business Process Layer invokes the Services Layer.

— The Services Layer binds and invokes Service Components in the Service Component Layer.

— Service Components in the Service Component Layer invoke Solution Components from the Operational and IT Systems Layer to carry out the service request.

— The response is sent back up to the service consumer.

7.7.8 Understanding ABBs

ABBs are the key building blocks of a particular layer.

An ABB is an abstraction that is associated with a capability and the outcome the capability will produce. To be precise, it is the implementation of ABBs that realize the capability and its outcomes in SOA solutions. The implementations constitute Solution Building Blocks (SBBs) that are "concrete", identifiable, and subject to governance and management.

When this part of ISO/IEC 18384 refers to ABBs 'providing', 'interacting', or being 'responsible' for, in fact, it is the implementation of the ABBs that is providing, interacting, or behaving in a solution. The simpler wording is often used for convenience; however, in practice, the interactions occur between ABB implementations.

ABBs are the components providing key capabilities that are expected from that layer. For example, the Integration Aspect is expected to provide meditation, routing, and protocol transformation capabilities. Therefore, these capabilities are realized by a set of solution building blocks, each of which provides exactly that atomic unit of architectural capability needed.

ABBs in the SOA RA and artefacts that are created as part of creating a solution are not always the same, For example, a process definition or a process model artefact is used by the Business Process ABB in the Process Layer to describe the underlying business process.

Some ABBs that are related to the functionality within an application such as

— Consumer Layer: Portal to loan processing application,

— Process Layer: Initiation of a loan processing application,

— Services Layer: Services that are required to support the loan processing application,

— Component Layer: Software components that need to be built that support the realization and implementation of the services, and

— Operational and IT Systems Layer: Actual runtime environment in which the components, legacy systems, all back-end applications, and packaged applications reside and run.

The cross-cutting layers, which include the Integration Aspect, serve as the central point where integration occurs across the enterprise and consolidates the design decisions into a set of software components that facilitate and enable the actual integration of applications.

The Information Aspect includes all data and information needed to support and instantiate each of the layers. Note that it is a cross-cutting layer indicating that each of the horizontal layers may have and will have data associated with their functioning and they will draw upon this data from the Information Aspect via metadata, actual data, or analytics.

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

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

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

7.7.9 Provisioning Services

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

7.7.10 Invoking Services

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

The Consumer Layer provides access to services. It allows consumers to consume services. The means of consumption of services can be a GUI or programmatic means of access to the services.

The Consumer Layer provides capabilities that support invoking the service endpoint. It accesses the services via the Integration Aspect or it can access the service directly if the given architecture does not wish to implement an Integration Aspect. Invocation is done by opening up access to the Consumer Layer.

7.7.11 Registries and Repositories

Registries and repositories may need to exist or be used in multiple layers during a physical implementation in many projects; therefore, the registry/repository is organized in the Governance Aspect of the SOA RA. In this way, it can be managed, monitored, and administered from a single logical location even though physically it may be federated or distributed. Logically, the functional layers and aspects may use the Registry/Repository ABB in the Governance Layer when needed.

7.7.12 Policies and Business Rules

Policy definition is the responsibility of the Governance Aspect. The definition of Policy Enforcement Points (PEPs) and policy enforcers is the responsibility of the individual horizontal/functional layers, such as Services Layer, Process Layer, etc. The enforcement of those policies is then the responsibility of a cross-cutting Management and Security Aspect where these capabilities have been consolidated. Therefore, monitoring and enforcement of the policies are the responsibility of the Management and Security Aspect while the administration of the policies remains the responsibility of the Governance Aspect.

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

7.7.13 Events

No single layer is responsible for events. Many layers and corresponding ABBs collaborate to produce a composite capability related to events such as Complex Event Processing (CEP), Business Activity Monitoring (BAM), etc. The cross-cutting nature of events is handled within the Integration Aspect, Management and Security Aspect, and Information Aspect. The building blocks for producing and listening for events reside within the Integration Aspect. The Management and Security Aspect is responsible for monitoring and managing the events. The Information Aspect is used to define the events.

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

Thus, CEP is addressed across multiple layers of the SOA RA; it may start from the Process Layer or Information Aspect.

7.7.14 Auditing and Logging

Enabling, auditing, and logging appropriately is important to the long term health of the SOA Solution. Auditing and logging enable compliance with regulations, resolution of problems, and provide a base of information about how the SOA Solution is running long term that can be analysed and mined using business analytics tools. In addition, these analytics can feed into governance of the SOA solution, providing key information on how well the SOA solution is continuing to meet business needs and enabling the business to choose to update the SOA solution.

Auditing and logging is done using capabilities from the Integration Aspect (logger and auditor ABBs), the management and security layer (Traceability Enabler/Auditor and Audit and Log ABB and logging manager).

Care should be taken when defining auditing and logging policies because there is a trading off between what will be audited and logged and resource consequences, e.g. effects on performance, network bandwidth, and storage availability.

7.7.15 Understanding different logical elements

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

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

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

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

Now let us look at the creation of a service and its conversion into a runtime entity. The service is created using an Integrated Development Environment (IDE) of some sort, with resulting Service, Technical, and Functional Components. Next, the Service, Functional, and Technical Components are bundled up into a Deployment Unit. Finally, the Deployment Unit is deployed into a runtime environment, becoming a Solution Component. Figure 6 shows the structural relationship between the different service-related ABBs.

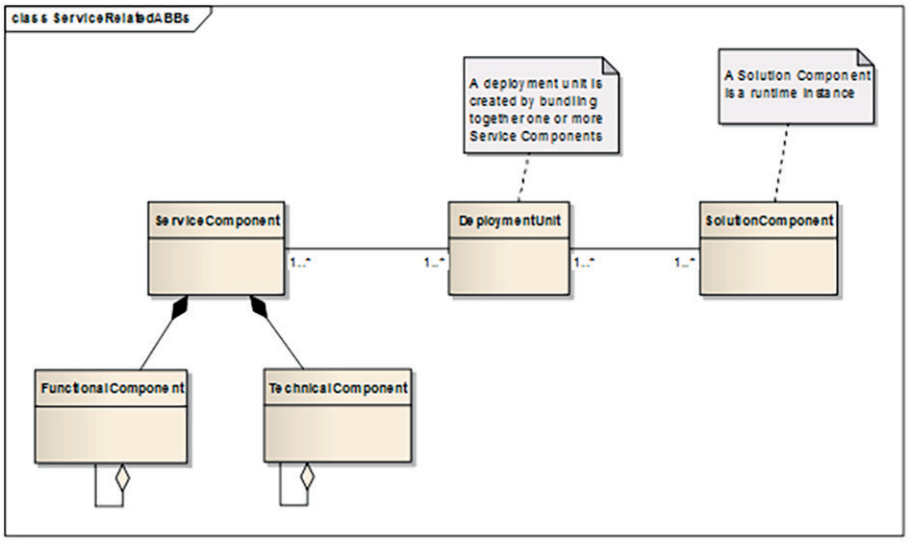


Figure 6 — Structural Relationship between Service-Related ABBs

Figure 7 shows the "lifecycle" or the dynamic relationship between the design and development time Service, Functional, and Technical Components, the bundled Deployment Unit, and the runtime Solution Component.

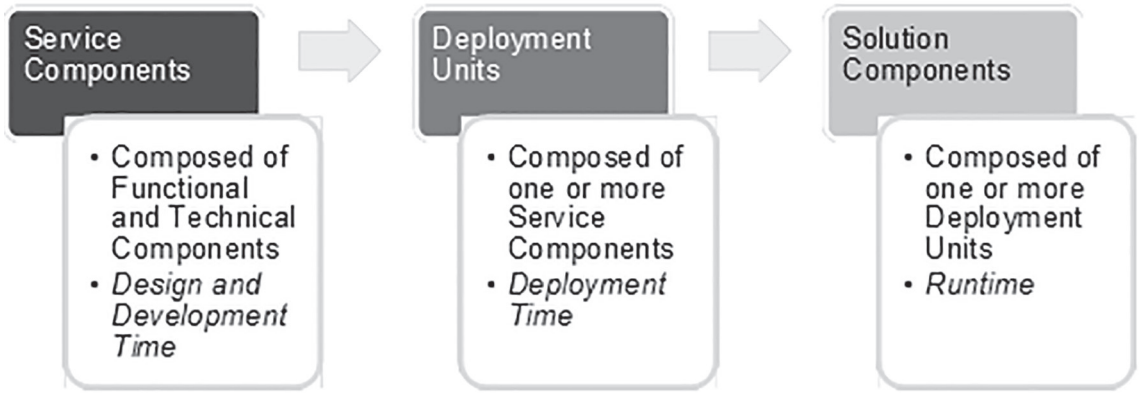


Figure 7 — Dynamic/Temporal Relationship between Service Components, Deployment Units and Solution Components

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

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

in the runtime environment (in the appropriate directory), the .dll component becomes the Solution Component for the CheckCredit service.

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

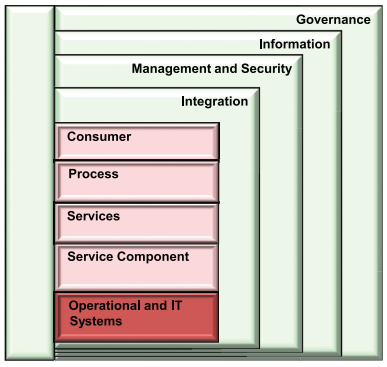
So what is a Solution Building Block? In both services considered in the example above, the service gets deployed into a Solution Building Block which provides the runtime implementation of a Service Container. In the case of the .NET service (CheckCredit), the Solution Building Block is (for the sake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

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

8 Operational and IT Systems Layer

8.1 Overview

8.1.1 Summary



(From 7.5.2) The Operational and IT Systems layer captures the new and existing organization infrastructure needed to support the SOA solution at design, deploy and run time. This includes the following:

— all infrastructures to run the SOA and its components;

— all operational and runtime hosting of components, both physical and infrastructural;

— all deployment time components;

— all resources to support services, data, and application systems;

— all assets required to support the functionality of the service in the SOA, where the assets may include custom or packaged application assets, new services, services created through composition or orchestration, infrastructure services, etc.

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 Aspects. 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.

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

— existing monolithic custom applications;

— existing transaction processing systems;

— existing databases;

— existing package applications and solutions including ERP and CRM packages;

— legacy applications and systems;

— access to existing web services;

— systems built from web services;

— IT Infrastructure;

— Enterprise Application Integration (EAI) systems;

— Service Resources providing access to existing business elements or third-party elements;

— Data Resources providing physical storage of data in the business solution;

— Application System Resources providing specific business functions;

— access to existing business services.

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 construct and the rest of the SOA in the wider context of Cloud computing. Key requirements for this layer are outlined in the capability clause describing the capabilities provided to fulfil those requirements.

8.1.2 Context and Typical Flow

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 a thought experiment is conducted which "freezes" the operations within this layer in a frame of time and expands it out, the separation of concerns that tend to cluster building blocks within the architecture will be discovered: the parts 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 five major cross-cutting and supporting Aspects (development, integration, information, Management and Security (MaS) factors, and governance).

Since this layer represents a point in time and logical categorization and generalization of the runtime environment, then 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 a capability for a SOA solution involves systems using mainframe and Java EE (see Reference [24]) platforms, necessary Architecture Building Blocks (ABBs) from the Integration Aspect

and Service Component Layer need to be instantiated using the underlying mainframe and Java EE components which provide the functional capability.

This formula can be expressed as:

Operational and IT 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)

8.1.3 Capabilities

There are multiple categories of capabilities that the Operational and IT Systems Layer needs to support. These categories of capabilities are as follows.

— **Service Delivery**: This category of capabilities is required for delivery of the functional elements of services. This includes the finding of the components implementing the services, the wrapping and the composition/decomposition of the underlying services, and the implementation of the services.

— **Runtime Environment**: This category of capabilities is required for providing a runtime environment representing runtime infrastructure for SOA. This includes capabilities to support both the components required to support service functionality and those required to actually run the components and building blocks of the SOA RA itself. This includes capabilities for the following:

  — the hardware, operating system components;

  — the Solution Building Blocks, which are the runtime instances or realizations of the ABBs of all layers in the SOA RA that have been selected for inclusion in a particular operating environment.

— **Virtualization and Infrastructure Services**: This category of capabilities provides underlying infrastructure such as computing power, network, storage, etc. in native or a virtualized manner.

This layer features the following capabilities in these categories.

— **Service Delivery**

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

— **Runtime Environment**

  11) Ability to support operating systems platforms
  12) Ability to support runtime hosting platforms
  13) Ability to support runtimes of software needed to run service implementation
  14) Ability to support runtimes and software needed to deploy service implementations

15) Ability to run supporting ABBs and Solution Building Blocks from other layers of the SOA RA

16) Ability to support the software environment in which the Solution Component runs

— **Virtualization and Infrastructure Services**

17) Ability to provide infrastructure needed by runtime infrastructure

18) Ability to provide infrastructure in a virtualized manner to platforms

19) Ability to provide infrastructure in a virtualized manner to service implementation

20) Ability to manage infrastructure and virtualized infrastructure

21) Ability to provide a single point of control for security of the Operational and IT Systems Layer

These capabilities can be used to group the ABBs in 8.1.4.

8.1.4 Structural Overview of the Layer

The ABBs in the Operational and IT Systems Layer are logically partitioned into the categories that support the following:

— Solution Components which provide the functional capability of services and the solution;

— the runtime environment required by the SOA solution and that runs the actual Solution Components and their supporting infrastructure;

— service delivery which provides interfaces with underlying infrastructure components, such that they can be virtualized and effectively leveraged by the architecture.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 8.1.3 that the ABB supports. For example, in Figure 8, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Integration Aspect is shown as black (with a prefix of "Integration"). Therefore, 'MaS: Policy Enforcer 'and it supports capability number '2: Ability to host applications and functionality to deliver service features' and 'Integration: Integration Controller' supports '5: Ability to act as a broker between service requests and invoking implementations.

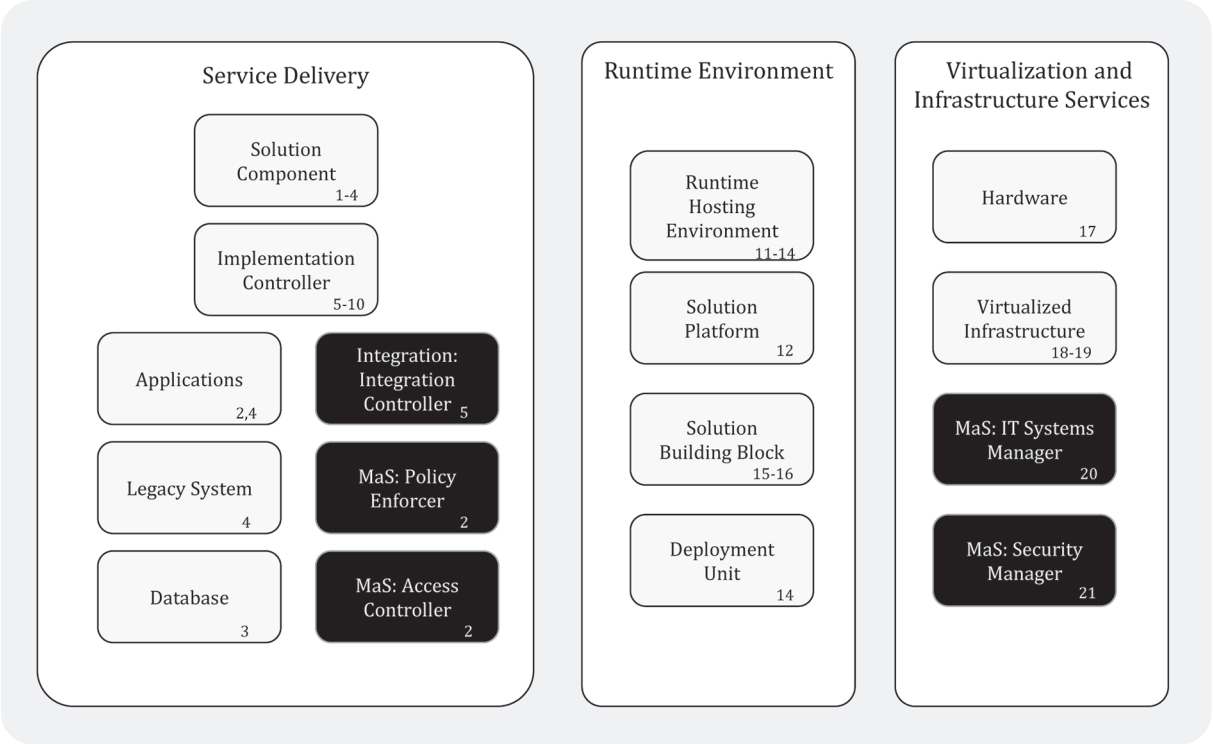


Figure 8 — ABBs in the Operational and IT Systems Layer

The details of the ABBs in 8.2 are grouped by the capabilities they support with ABBs from other layers listed last.

8.2 Details of ABBs and Supported Capabilities

8.2.1 Service Delivery

8.2.1.1 Solution Component

This ABB represents realization of systems 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 systems. For example, it could be an invocation of a legacy component, or an existing or new database request, application, or a component wrapped within a Commercial Off-The-Shelf (COTS) package. Thus, it is a runtime instantiation of a group of cohesive components residing within a system that collectively provide an implementation for a set of related services.

8.2.1.2 Implementation Controller (IC)

This ABB represents elements that receive 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 Component. This is required because historically, most legacy applications have not been written with the intent of being elements in an SOA and the service Solution Components within them need to be exposed through service composition and decomposition.

8.2.1.3 Applications (Packaged and Custom)

This ABB represents the applications and operations performed on them that are running as units of execution within the runtime environment of an SOA solution, for example, existing human resources systems, SAS systems, and other domain specific systems.

8.2.1.4 Legacy system

This ABB represents the legacy systems and operations performed on them in an SOA solution.

8.2.1.5 Database

This ABB represents the databases and operations performed on them in an SOA solution.

8.2.1.6 Integration Aspect: Integration Controller

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

8.2.1.7 Management and Security Aspect: Policy Enforcer

See 14.2.7.1.

8.2.1.8 Management and Security Aspect: Access Controller

See 14.2.2.8.

8.2.2 Runtime Environment

8.2.2.1 Runtime Hosting Environment (RHE)

This ABB represents elements that provide 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.

8.2.2.2 Solution Platform

This ABB represents elements that support the software environment in which the solution components and solution building blocks deploy and run. Examples would be Java virtual machines (JVMs) hosting a service container solution building block or a CICS environment.

8.2.2.3 Solution building block

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

8.2.2.4 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. The instantiation of this ABB is deployed on Solution Platforms.

8.2.3 Virtualization and Infrastructure Services

8.2.3.1 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).

8.2.3.2 Virtualized Infrastructure

This ABB represents elements that support 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.

8.2.3.3 Management and Security Aspect: IT Systems Manager

See 14.2.3.1.

8.2.3.4 Management and Security Aspect: Security Manager

See 14.2.2.1.

8.3 Inter-Relationships between the ABBs

The Solution Component ABB represents a key ABB interacting with other ABBs. Since it contains the Service Component, Functional Components, and Technical Components from the Service Component Layer for realizing services, it interacts with the implementation controller and other ABBs as needed. Requests are first validated as secure by the Access Controller ABB and Security Manager ABB in the Management and Security Aspect, and then translated into Solution Component ABB requests by the Implementation Controller ABB and executed by the Solution Components in the Solution Platform.

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

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

In Figure 9, the arrows between the ABBs indicate an interaction from one ABB to another.

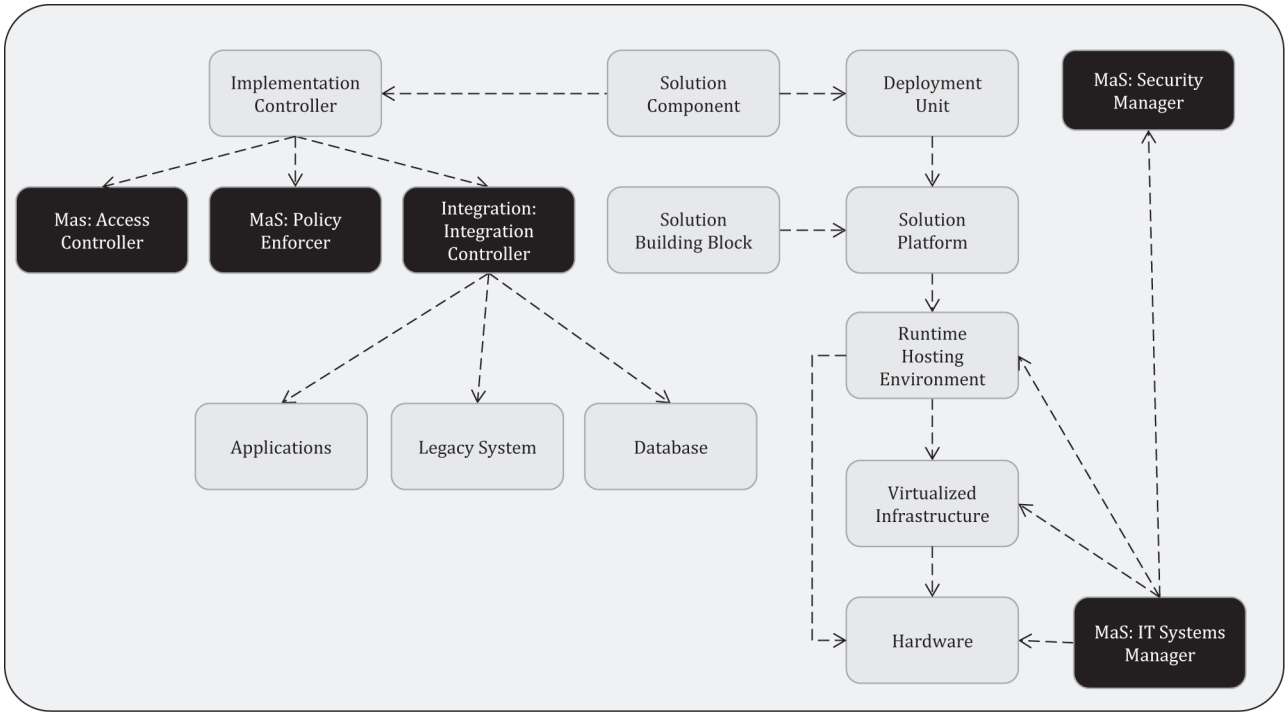


Figure 9 — Relationships among ABBs in the Operational and IT Systems Layer

8.4 Significant Intersection Points with other Layers

8.4.1 General

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

This connection is summarized in 8.3 in terms of intersection with the rest of the SOA RA including cross-cutting aspects.

8.4.2 Intersection with the rest of the SOA RA

The following are the two points of intersection of the Operational and IT Systems Layer with the rest of the SOA RA:

a) The first perspective is that the Operational and IT 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 from the Service Component Layer, and 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 system;

b) The second perspective is that the Operational and IT 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.

8.4.3 Interaction with Cross-Cutting Aspects

The Operational and IT Systems Layer relies on cross-cutting Aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

It relies on the Development Aspect for the following capabilities:

— ability to implement and test resources and connectors to Service Components with tools;

— ability to process deployment descriptions and hosting environment descriptions in order to deploy supporting resources into the Runtime Hosting Environment;

— ability to perform performance load testing and simulation to optimize resource usage.

It relies on the Management and Security Aspect for the following capabilities:

— ability to authenticate/authorize for service invocation;

— ability to enforce operational policies;

— ability to monitor health and well-being of underlying infrastructure and solution and applications deployed on the infrastructure.

It relies on the Information Aspect for the following capabilities:

— ability to store and retrieve metadata and data.

It relies on the Integration Aspect for the following capabilities:

— ability to invoke business processes and/or services;

— ability to transform data from one format to another.

It relies on the Governance Aspect for the following capabilities:

— ability to set and store business rules;

— ability to manage policies for IT management;

— ability to manage security policies.

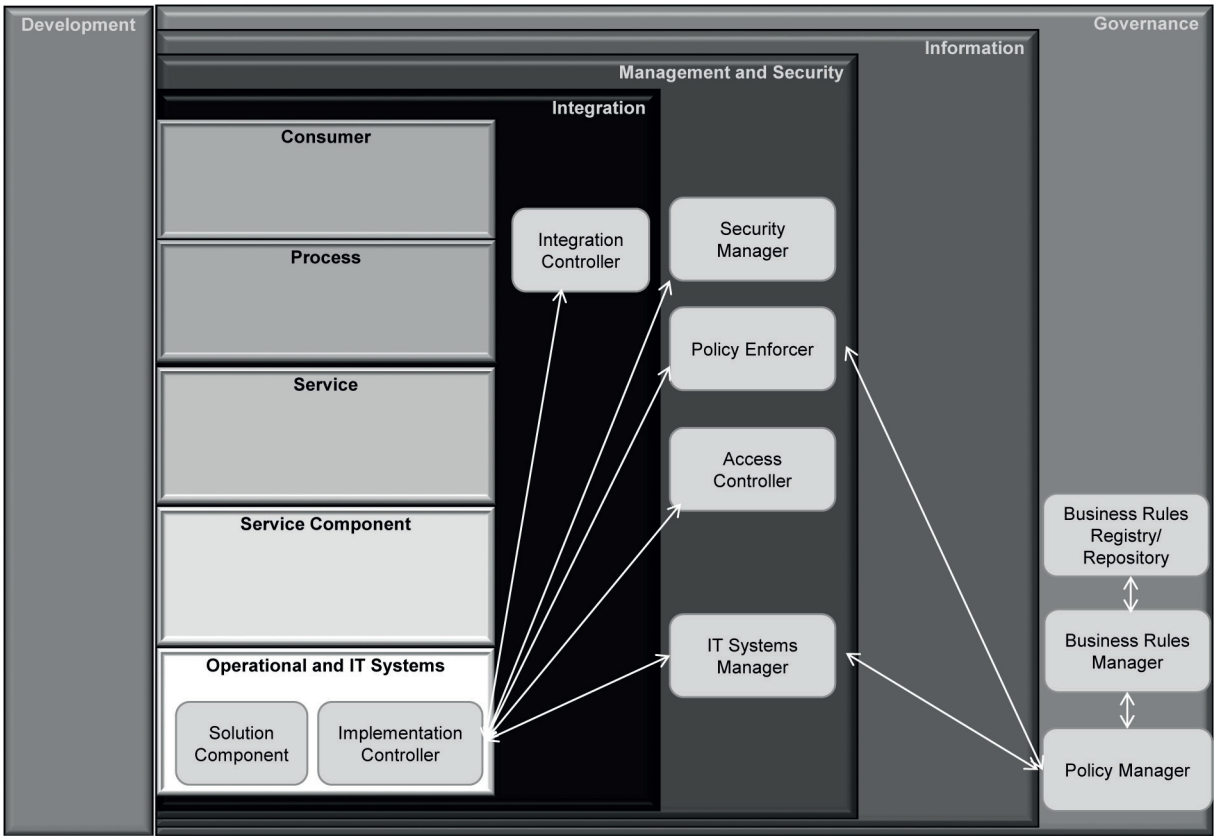


Figure 10 — Key Interactions of Operational and IT Systems Layer with Cross Cutting

Therefore, Operational and IT Systems Layer interfaces with the following ABBs of cross-cutting aspects of the architecture to provide its capabilities.

— It leverages Policy Manager ABB in the Governance Aspect 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.

— It leverages Access Controller ABB in the Management and Security Aspect to enforce access privileges and Policy Enforcer ABB in the Management and Security Aspect to enforce policies. These ABBs from the Management and Security Aspect enable the Operational and IT Systems Layer to operate across platforms and support a consistent set of policies for specific scenarios and limit the amount of associated risk. The Access Controller ABB and Policy Enforcer ABB in the Management and Security Aspect provide a single Policy Enforcement Point (PEP) for control for security for the Operational and IT Systems Layer and in practice for all the runtime components of the SOA RA. The policy enforcement could be federated. The Security Manager ABB in the Management and Security Aspect 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.

— It leverages IT Systems Management-related ABBs in the Management and Security Aspect 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.

— It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Aspect 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.

8.4.4 Interaction with Horizontal Layers

The Operational and IT 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, Process Layer, Services Layer, and Service Component Layer, has some ABBs specific to the scope of the layer and some ABBs from the Aspects that are needed to provide a runtime environment for the functional parts of the solution.

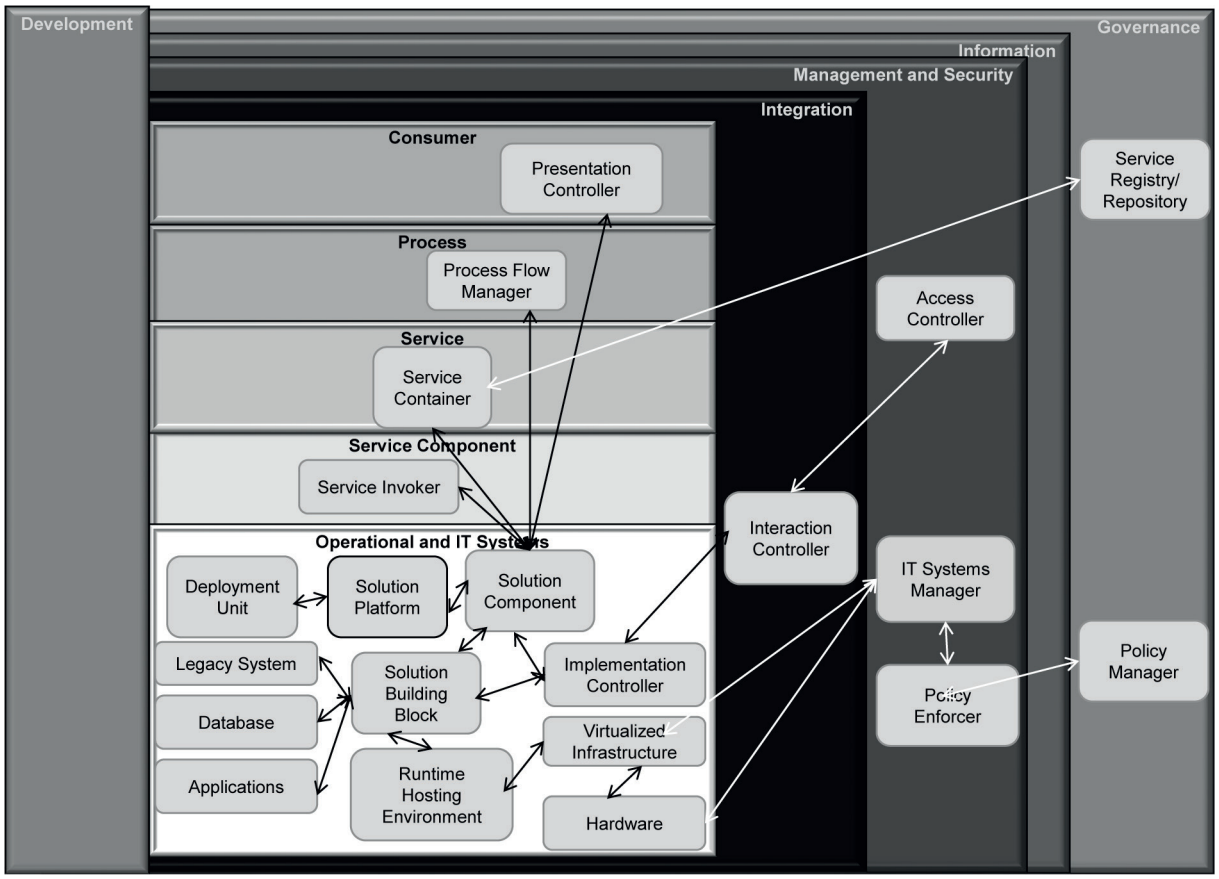


Figure 11 — Key Interactions of Operational and IT Systems Layer with Horizontal Layers

8.5 Usage Implications and Guidance

8.5.1 Options and Design Decisions

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

From an SOA perspective, the Operational and IT 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. This integration in a perimeter-less, cross-organizational fashion enables the foundation for service re-use by allowing the sharing of functionality and supporting capabilities across the portfolio.

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

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

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

— existing monolithic custom applications including Java EE (see Reference [24]) and .NET (see Reference [23]) applications;

— existing SOA services;

— legacy applications and systems;

— existing transaction processing systems;

— existing databases;

— existing package applications and solutions including ERP and CRM packages.

8.5.2 Implementation Considerations

Considerations when using this layer include the following.

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

— Try to federate security, and potentially event monitoring, to support the traceability and the agility required for an effective SOA. For example, if Java EE environment is currently used and has built federation in, as the organization goes through a merger and acquisition scenario where the CICS, .NET, and SaaS components need to be added, this core security framework will be critical for incorporating these components in an agile manner.

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

  — isolation and containment services:

    — multi-tenancy;

    — data privacy (both data in motion and at rest);

    — auditability;

    — authorization/authentication/access control;

  — application support services:

    — elasticity – dynamic resource provisioning/de-provisioning;

    — service location awareness;

    — infrastructure QoS management (as opposed to Application Service QoS);

  — data integrity services:

    — disaster recovery;

    — high availability clustering;

    — data backup;

    — data QoS management;

- data mobility;
- infrastructure accounting services:
  - chargeback services;
  - configuration management/auditability;
  - capacity management services.

8.5.3 Runtime and Deployment View of the SOA RA

As described in the first paragraph, the Operational and IT Systems Layer supports all the capabilities of the infrastructure that are needed for running/executing all software. Therefore, this layer supports the execution of the capabilities and responsibilities of the other layers of the SOA RA, including the components implementing a service itself and the components that provide the SOA capabilities like Service Container ABB from the Services Layer, Data Transformer ABB from the Integration Aspect, Process Flow Manager ABB from the 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 Aspect runs as a Solution Building Block in the Operational and IT Systems Layer but a business service, such as Get Customer Information, runs ultimately as a Solution Component.

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

In Figure 12, this deployment view of the SOA RA is visualized in this UML component diagram.

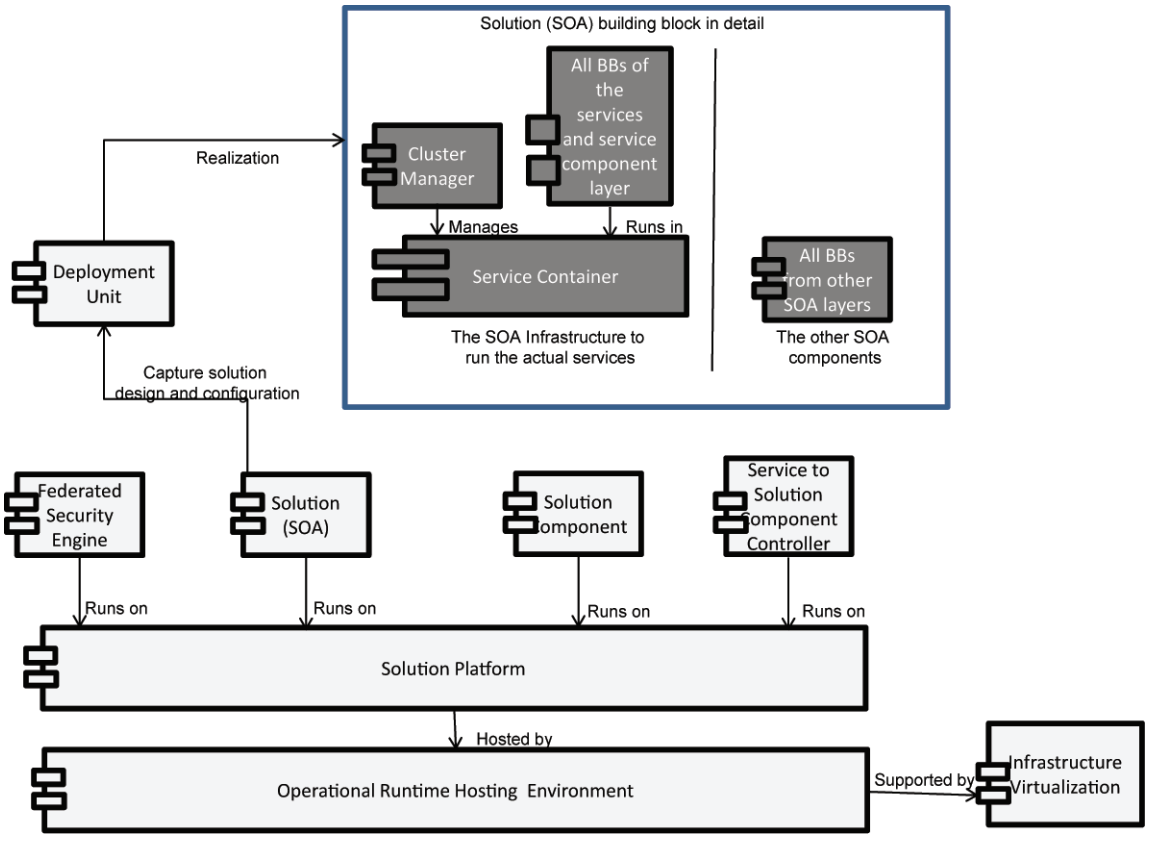


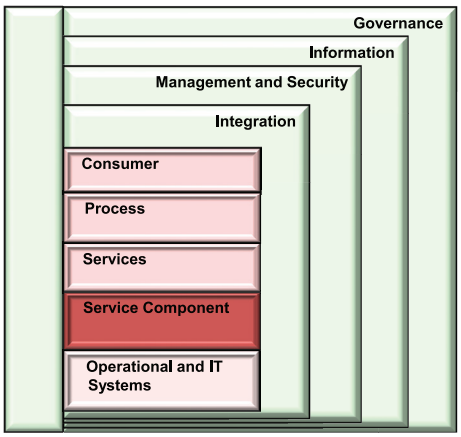
Figure 12 — Deployment View of the SOA RA

The light grey coloured building blocks are from the Operational and IT Systems Layer used to create a logical design. The deployment unit ABB is used to capture the design, implementation requirements, configuration, etc. and inform the instantiation of the architecture with real implementations of the ABB, solution building blocks (dark grey boxes).

9 Service Component Layer

9.1 Overview

9.1.1 Summary



(From 7.5.3) The Service Component Layer contains capabilities that support software components which represent the implementation or "realization" for services or operation on services; hence, the name Service Component. The layer 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 management and quality of service interactions. They "bind" the service interface to the implementation of the service in the Operational and IT Systems layer. Service components may be hosted in containers, as defined in the Services Layer, which support a service description (see 12.2.2.1 for more information on containers).

The Service Component Layer manifests the IT conformance with each service interface defined in the services layer; it helps guarantee the alignment of IT implementation with service description.

Each Service Component

— realizes one or more services,

— provides an enforcement point for "faithful" service realization (ensures quality of service and service level agreements),

— enables IT flexibility by strengthening the decoupling in the system by hiding volatile implementation details from service consumers,

— offers a façade behind which technologies can be deployed as required to enable service functionality, and

— generally contains business-specific logic with no reference to integration logic.

The Service Component Layer enables flexibility through encapsulation and by enabling loose coupling. A separation of concerns is achieved such that the service consumer can assume that the realization of the service is faithful to its published description (service compliance) and the service provider will ensure that such compliance is achieved. The details of the service realization are of no consequence to the consumer. The service provider is consequently able to replace one component with another having the same interface, producing the same outcomes (real world effects), and having identical conditions of use without any impact on service consumers.

9.1.2 Context and Typical Flow

The Service Component Layer provides the following, including

— ability to support the exposure of a service in a standards-compliant manner supporting interoperability; note that the protocol (SOAP/REST/Java EE, etc.) is not prescribed but determined by the associated architectural decision,

— ability to expose the service via an integration stack from the underlying platform in which the service functionality resides (a.k.a within the Operational and IT Systems Layer), and

— ability to publish and deploy the Service Component itself including

  — exposing services in an interoperable manner,

  — binding to the Operational and IT Systems Layer at runtime,

  — publishing service description information in an interoperable and standards-compliant manner so that other elements of the SOA can invoke it, and

  — deploying the service into the associated services container.

9.1.3 Capabilities

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

— **Service Realization and Implementation**: This category of capabilities supports the realization of the services.

— **Service Publication and Exposure**: This category of capabilities supports service exposure and service description publication.

— **Service Deployment**: This category of capabilities supports service deployment.

— **Service Invocation**: This category of capabilities supports service invocation.

— **Service Binding**: This category of capabilities supports service binding.

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

This layer features the following capabilities in these categories.

— **Service Realization and Implementation (Design Time)**

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

— **Service Publication and Exposure (Design Time)**

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

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

— **Service Deployment (Design Time)**

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

— **Service Invocation (Runtime)**

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

— **Service Binding (Runtime)**

6) Ability to support service interoperability

7) Ability to implement a part of the broker pattern, a.ka. an architectural pattern where a broker component coordinates communication

8) Ability to convert from the service description to service calls supported by the platform (in the case of a WSDL web service, the conversion from WSDL service description to the service invocation)

9) Ability to convert at runtime into a standards-compliant form for consumption by standards-compliant service consumers on both input and output

10) Ability to convert from the standards-compliant form to the form acceptable to the underlying Solution Component which satisfies the service's functional capability on both input and output

11) Ability to enforce policies and access control during service binding

9.1.4 Structural Overview of the Layer

The Service Component Layer can be thought of as supporting capabilities dealing with design-time and runtime concerns. One of the key responsibilities of the Service Component Layer is to provide the integration between other SOA RA layers (e.g. Integration Aspect) and the underlying Operational and IT 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 and IT Systems Layer. This binding is achieved through the implementation of the broker pattern.

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

— realization and implementation of services,

— exposure and contract publication of services,

— deployment of services,

— invocation of services, and

— binding of services.

Figure 13 illustrates the ABBs supporting the capabilities of the Service Component Layer.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 9.1.3 that the ABB supports. For example, in Figure 13, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Governance Aspect is shown as lighter grey (with a prefix of "Governance"). For example, in Figure 13, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. Therefore, MaS: Policy Enforcer supports capability number 11: Ability to enforce policies and access control during service binding and Governance: Service Registry/Repository supports capabilities 2 and 3.

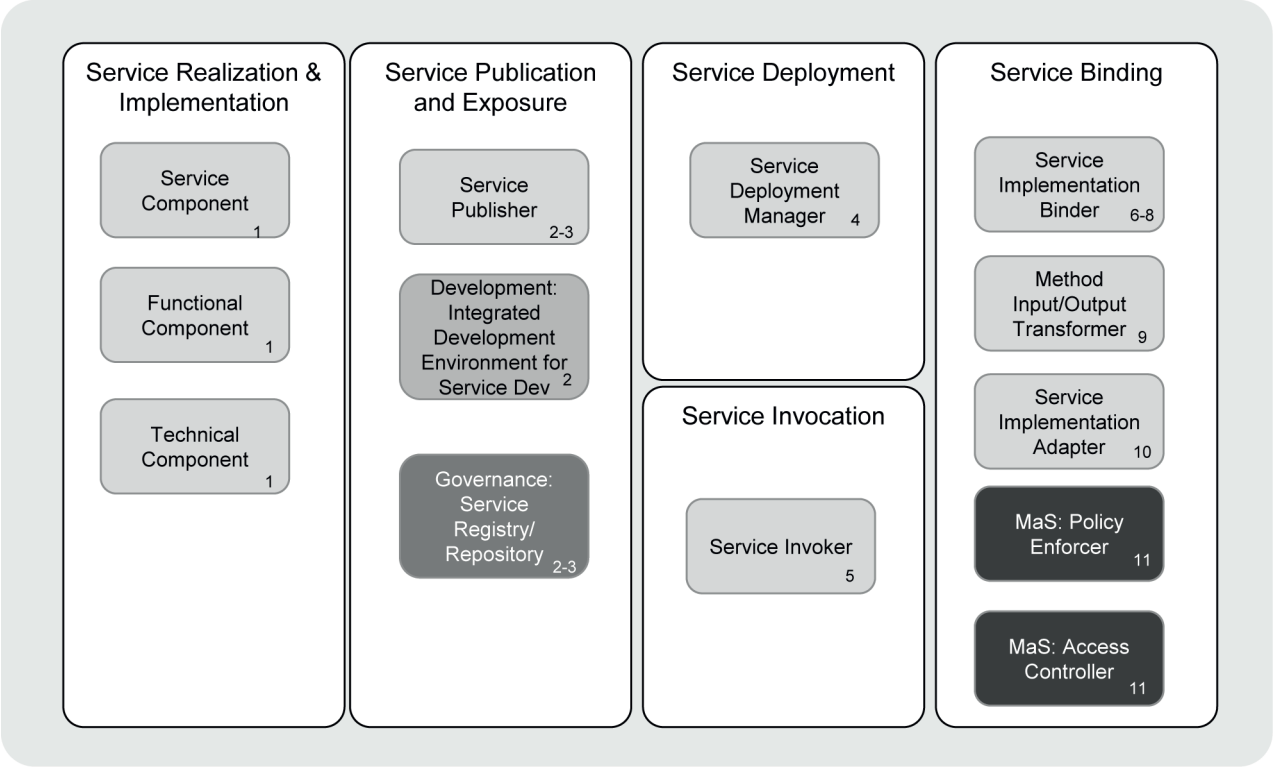


Figure 13 — ABBs in the Service Component Layer

The details of the ABBs in 9.2 are grouped by the capabilities.

9.2 Details of ABBs and Supported Capabilities

9.2.1 Service Realization and Implementation

9.2.1.1 Service Component

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

Service components reflect the definition of the service they realize, both in its functionality and its management and quality of service interactions. They "bind" the service description to the implementation of the service in the Operational and IT Systems Layer. Each Service Component provides an enforcement point to ensure quality of service and service level agreements. As a best practice, the Service Component should contain business-specific logic with no reference to integration logic. Service Components are composed of Functional and Technical Components which are all are logical, design-time assets. Service, Functional, and Technical Components are bundled up into a Deployment Unit. Finally, the Deployment Unit is deployed into a runtime environment, becoming a Solution Component.

9.2.1.2 Functional Component

This ABB represents 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. This could be a legacy system invocation or one or more other Functional Components.

9.2.1.3 Technical Component

This ABB represents an abstraction of infrastructure to support Functional Components. The Technical Component encapsulates the technical capabilities to support standards compliance and technical support that a service provides.

9.2.2 Service Publication and Exposure

9.2.2.1 Service Publisher

This ABB represents capabilities to publish Service Component design-time metadata and description to a design-time Service Registry/Repository ABB in the Governance Aspect for use by other SOA layers and cross-cutting Aspects. The Service Publisher may invoke 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.

9.2.2.2 Development: Integrated Development Environment (IDE) for Service Development

See 17.2.3.8.

9.2.2.3 Governance Aspect: Service Registry/Repository

See 16.2.2.3.

9.2.3 Service Deployment

9.2.3.1 Service Deployment Manager

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

9.2.4 Service Invocation

9.2.4.1 Service Invoker

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

9.2.5 Service Binding

9.2.5.1 Service Implementation Binder

This ABB represents 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 Aspect component would require its service request converted or mapped to an underlying Service Component call. The converting the in-bound call leverage Method Input/Output Transformer ABB is the responsibility of this ABB.

9.2.5.2 Method Input/Output Transformer

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

9.2.5.3 Service Implementation Adapter

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

9.2.5.4 Management and Security Aspect: Policy Enforcer

See 14.2.7.1.

9.2.5.5 Management and Security Aspect: Access Controller

See 14.2.2.8.

9.3 Inter-Relationships between the ABBs

As mentioned earlier, the ABBs in the Service Component Layer support the design-time and the runtime capabilities of the layer.

Figure 14 illustrates one of many different potential interaction flows among ABBs in the Service Component Layer enabling design-time capabilities.

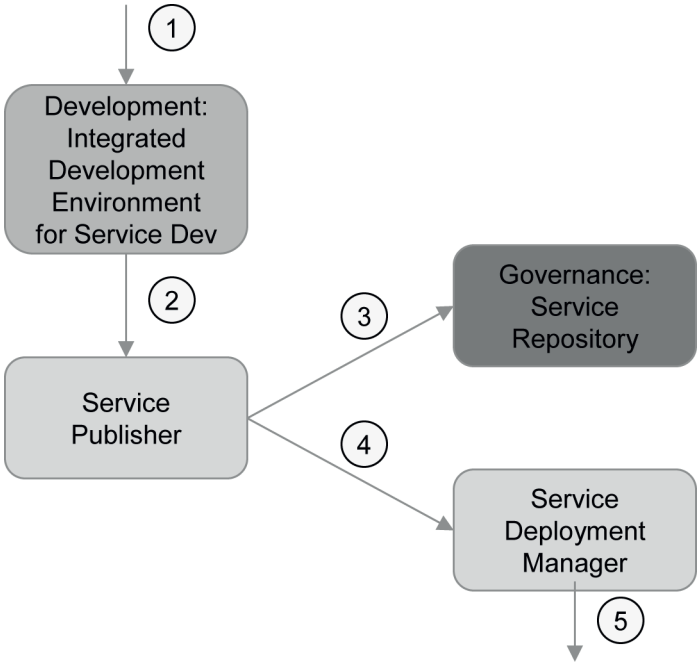


Figure 14 — Illustrative Interaction Flow among Design-Time ABBs in the Service Component Layer

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

a) During service design, an Integrated Development Environment (IDE) for Service Development is used to develop a service contract and description using agreed upon standards.

b) The service contract and description is given to a Service Publisher ABB to publish the service.

c) The Service Publisher ABB publishes the service contract and description in the Service Registry/Repository ABB in the Governance Aspect for use by other SOA layers and cross-cutting Aspects.

d) The Service Publisher also invokes a Service Deployment Manager to execute deployment functions so that the service is available to access in the appropriate services container, according to the contract and other information.

Figure 15 illustrates the interaction flows among ABBs in the Service Component Layer enabling runtime capabilities.

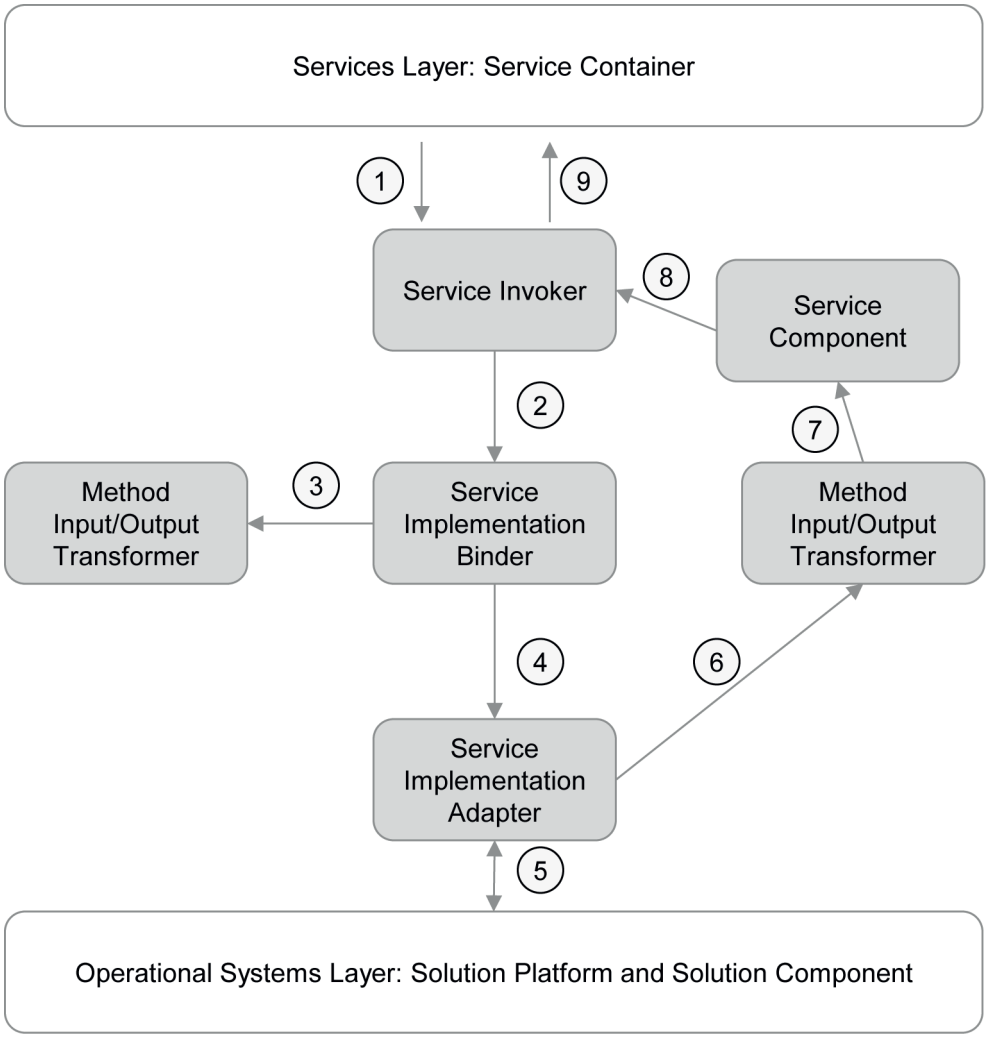


Figure 15 — Illustrative Interaction Flow among Runtime ABBs in the Service Component Layer

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

a) The Service Invoker ABB is invoked from all other layers of the SOA RA (except the Operational and IT Systems Layer) and provides the ability for the Services Layer to invoke the Service Component realizing the services.

b) The Service Invoker ABB calls the Service Component ABB which is the binding stack to bind to external layers.

c) 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.

d) The Service Component ABB then passes control to the Service Implementation Adapter.

e) The Service Implementation Adapter then maps the invocation into the Operational and IT Systems Layer.

In returning, the response to the method invocation to the consumer, the Method Input/Output Transformer can be invoked to change the response message appropriately before returning the response to the service invoker which will send the response to the consumer.

9.4 Significant Intersection Points with other Layers

9.4.1 General

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 and IT Systems Layer with service description. Each Service Component

— provides an enforcement point for "faithful" service realization (ensures QoS and Service-Level Agreements (SLAs),

— enables business flexibility by supporting the functional implementation of IT flexible services, their composition, and layering, and

— enables IT flexibility by strengthening the decoupling in the system; decoupling is achieved by hiding volatile implementation details from consumers.

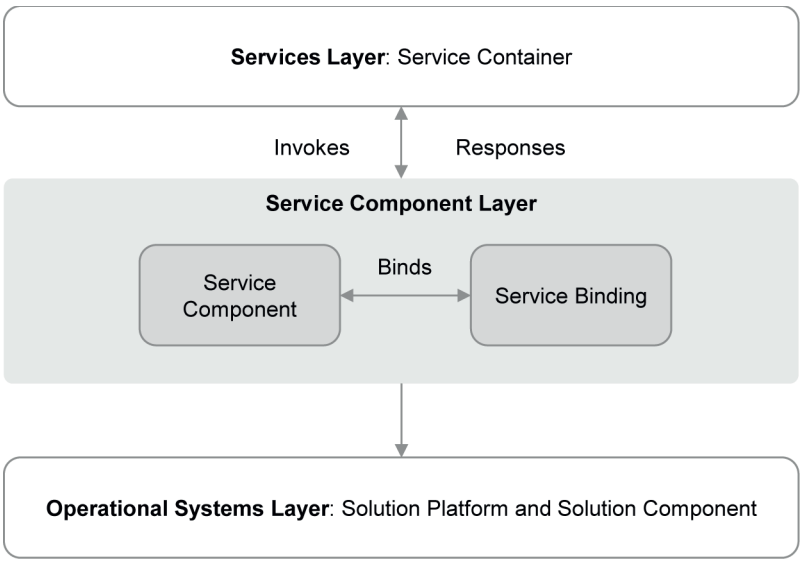


Figure 16 — High-Level Interaction of the Service Component Layer with Layers Above and Below in the SOA RA

As shown in Figure 16, the Solution Component ABB in the Operational and IT Systems Layer can be thought of as a runtime instantiation of a solution enabling a system of services. A system is implemented by one or more Service Components realizing one of more services and related Functional and Technical Components. The Solution Component ABB is a runtime instantiation of the Service Components and associated functional and technical components realizing a system of services. Service architects and developers determine which standards to conform to in the protocols for the service, as well as to connect to the underlying Operational and IT Systems Layer. Which standards are used to describe the service (e.g. WSDL) and these protocols is also an important decision.

9.4.2 Interaction with Cross-Cutting Aspects

The Service Component Layer relies on cross-cutting Aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

It relies on the Development Aspect for the following capabilities:

— ability to implement and test Service Components with tools;

— ability to process service descriptions, contracts, and deployment descriptions in order to deploy the appropriate component and enable publication of the service description;

— ability to use descriptions in integrated development environments to create the appropriate service implementation.

It relies on the Governance Aspect for the following capabilities:

— ability to store metadata about services during design time;

— ability to define and manage (storage, retrieval, etc.) rules used by the components realizing the services.

It relies on the Management and Security Aspect for the following capabilities:

— ability to authorize during invocations on the underlying components.

It relies on the Information Aspect for the following capabilities:

— ability to store and retrieve metadata and data required by the components.

It relies on the Integration Aspect for the following capabilities:

— ability to transform data from one format to another.

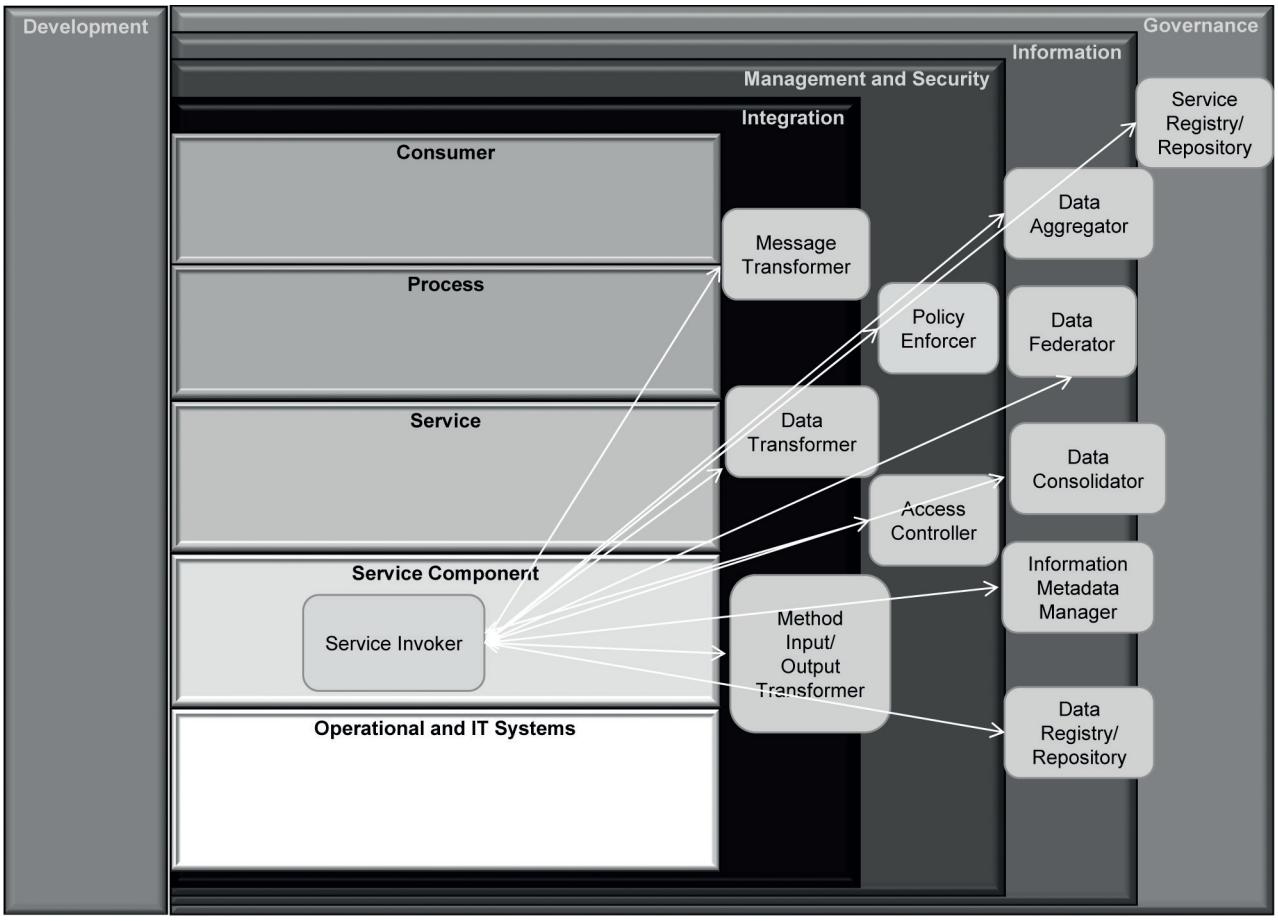


Figure 17 — Key Interactions of the Service Component Layer with Cross-Cutting Aspects

Figure 17 shows the Service Component Layer interfaces with the following ABBs of cross-cutting aspects of the architecture to provide its capabilities.

— It leverages the Access Controller ABB and Policy Enforcer ABBs in the Management and Security Aspect to enforce access control privileges and other policies.

— It leverages the Data Aggregator ABB, Data Federator ABB, Data Consolidator ABB, Information Metadata Manager ABB, and Data Registry/Repository ABB from the Information Aspect to provide information about services for the other layers of the SOA RA.

— It leverages the Message Transformer ABB and Data Transformer ABB from the Integration Aspect to transform data from one format to another. The Method Input/Output Transformer ABB 2280 leverages these ABBs from the Integration Aspect.

— It leverages the Service Registry/Repository ABB in the Governance Aspect to store metadata about services. IT Governance has a significant influence on the Service Component Layer. The choice of implementation technology, the manner in which Service Components may/may not consume behaviours from other Service Components, and decisions about where to place integration logic are examples of cases in which this layer may be influenced by IT Governance and the Governance Aspect. 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.

9.4.3 Interaction with Horizontal Layers

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

— The Service Invoker ABB interacts with the Services Layer and Integration Aspect.

— The Service Publisher ABB interacts with the Services Layer.

— The Service Deployment Manager ABB interacts with the Operational and IT Systems Layer.

— The Service Implementation Adapter ABB interacts with the Operational and IT Systems Layer.

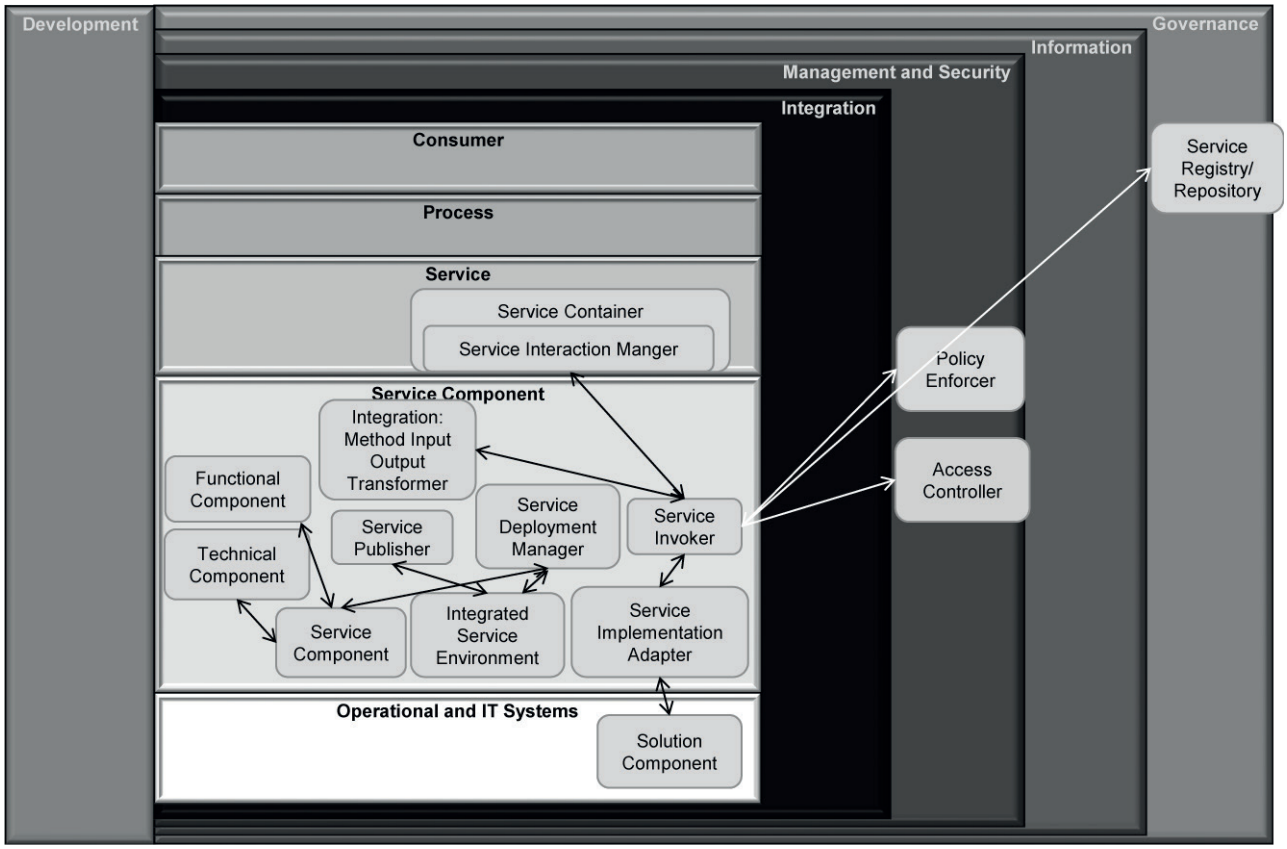


Figure 18 — Key Interactions of the Service Component Layer with Horizontal Layers

9.4.4 Interaction with the Services Layer

By its nature, this layer is coupled to the Services Layer of the SOA RA. A service definition change is likely to cause a direct side-effect on the Service Component in this layer. For example, if a service is retired from the Services Layer, the corresponding Service Component could also be retired if no other services are using it.[2)] Finally, the Service Components reflects the definition of one or more services. To ensure that this relationship is maintained, a Service Component should not exhibit behaviours not defined in a service description.

2) A service component can be used by other components and services in the solution, so when one service is retired, some of its components may still be in use by other services. It is important to maintain the connection of the components with the services and the governance regimen.

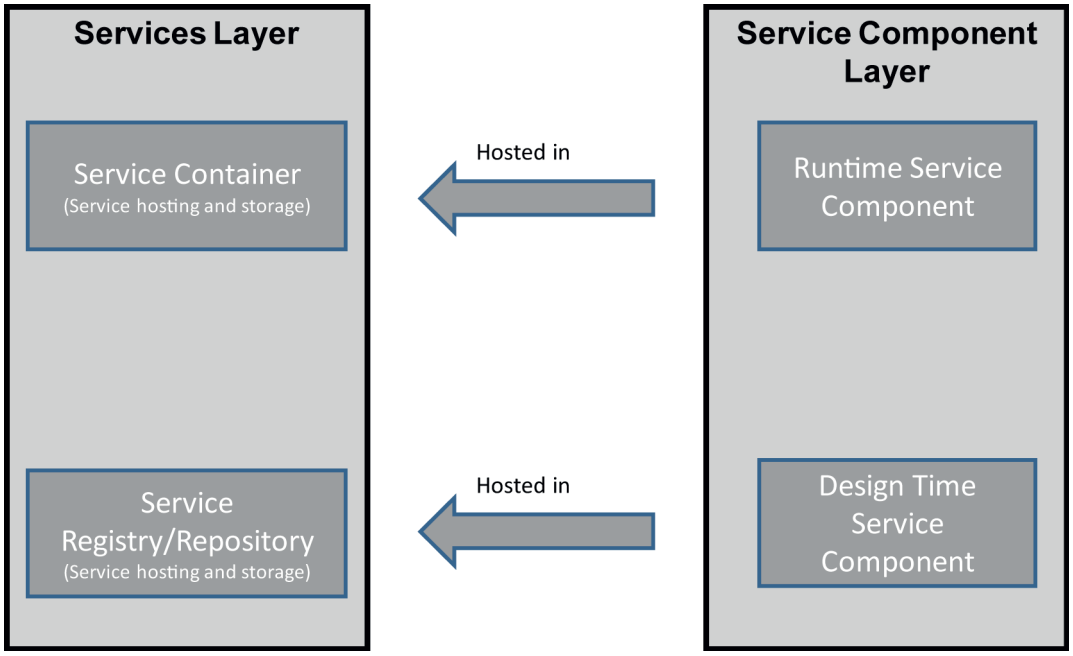


Figure 19 — Relationships between the Services Layer and Service Component Layer

The runtime relationship between the Service Component Layer and the Services Layer is illustrated in Figure 20. Services are deployed in the services container in the Services Layer. The services can be discovered using the Service Registry/Repository ABB in the Governance Aspect; this can provide the contract and support for virtualization. The Service Integration Manager for the service then invokes the corresponding Service Component in the Services Component Layer which is then bound to the solution platform and invoked in the Operational and IT Systems Layer.

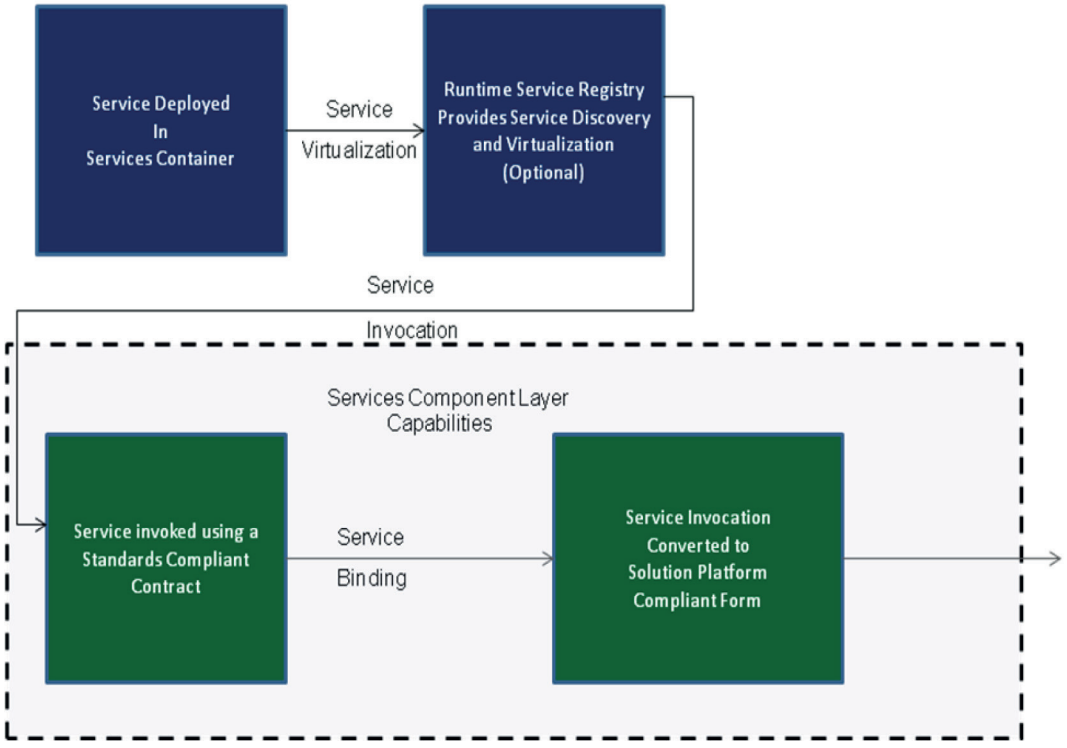


Figure 20 — Use of Runtime Capabilities in the Service Component Layer

9.4.5 Interactions with the Operational and IT Systems Layer

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

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

9.5 Usage Implications and Guidance

9.5.1 Options and Design Decisions

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

— **Capability**: The capability to realize the value proposition of Service Components and to realize the required behaviour of a given service.

— **Familiarity**: Whether the capability of using the technology already exists in the organization.

— **Strategic**: Whether the technology is in line with the technology roadmap of the organization.

— **Manageability**: Whether the technology allows for the effective management of Service Components with respect to the Key Performance Indicators (KPIs) defined.

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

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

— What is the best hosting environment for a particular application?

— What is an appropriate runtime environment for a particular subset of an application?

— What kind of runtime capabilities are required in terms of NFRs?

— Consider different integration patterns, both traditional and service oriented. Should integration always occur through a Mediator ABB in the Integration Aspect? Can it be performed inside a Service Component?

— Should traditional software integration be used or the service paradigm be used, i.e. do I make it a service, for example, should collaborating Service Components consume each other through the Services Layer or through a platform-specific interface (e.g. EJB-to-EJB or EJB-to-service)?

— Where are transformations performed? In the Service Component Layer to support specific legacy systems or in the Integration Aspect to support general mediation?

— Should component implementations be portable across multiple runtime environments?

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

Latency refers to the delay of accessing a specific service due to internal implementation details and processes. Availability refers to the percentage of how a specific service can be available during a specific 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 an authorization and authentication facility to ensure secure access.

9.5.2 Implementation Considerations

9.5.2.1 Typical Interaction Sequences: A Narrative of the Flows

Refer to the example illustrated in Figure 21 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 assumes 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 description defined in the interface in Service A.

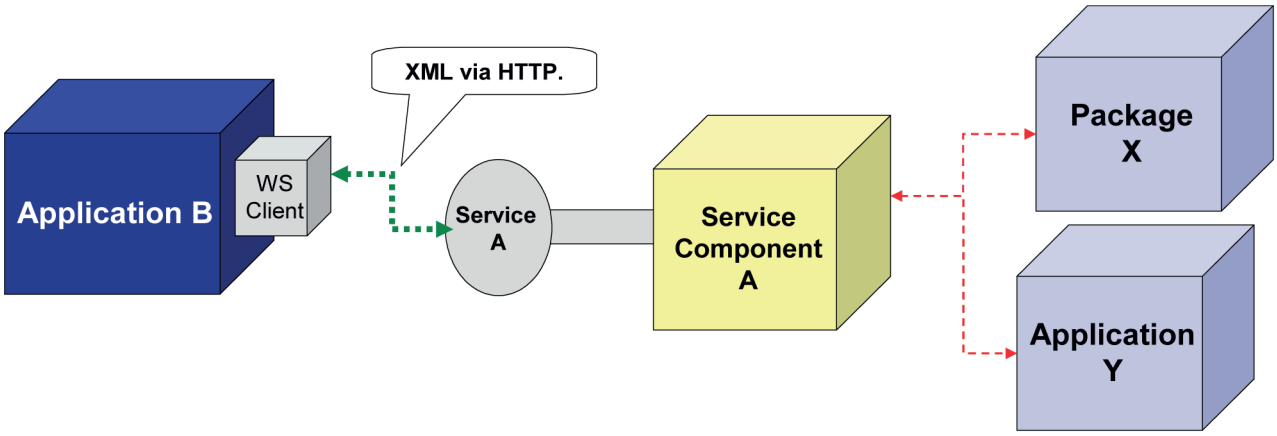


Figure 21 — Service Components as a Facade

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, assuming the substituted packages are equivalent and have the same outcomes. 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 occurs frequently in the context of transforming legacy systems into services. Figure 22 shows an example of how to construct Service Components using existing application assets. Assume two existing application systems, Application 1 that maintains customer addresses and Application 2 that validates postal codes. These two existing applications were developed on proprietary platforms using proprietary technologies. In other words, these two applications are legacy systems. A new project intends to build two web services that can be accessed

via the SOAP protocol. 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.

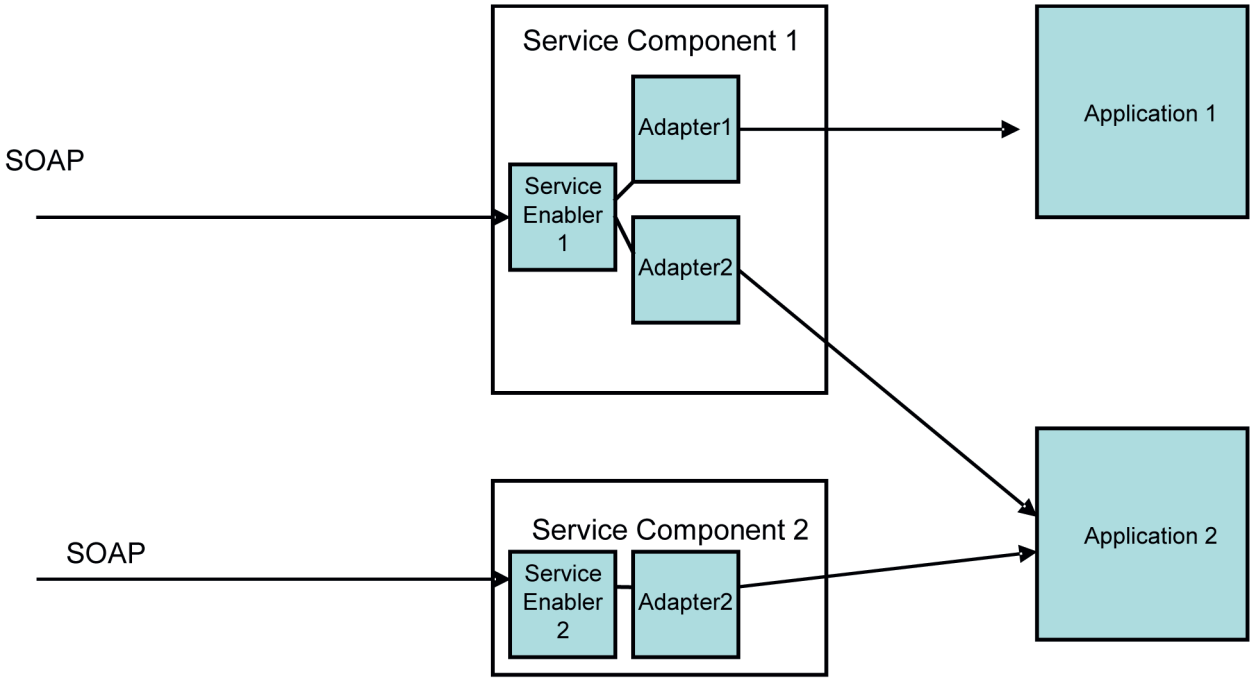


Figure 22 — Interaction Flow in a Composition Scenario

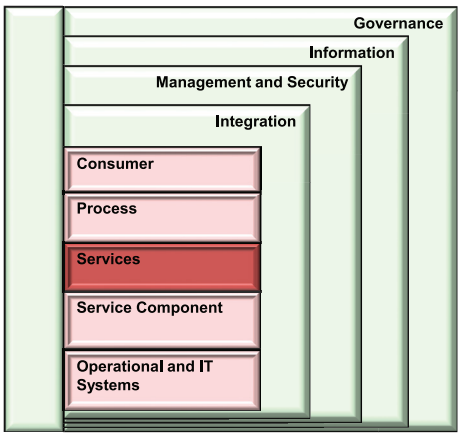
As shown in the example in Figure 22, there is a need to have an “adapter” component that represents particular existing application assets and provides an API for Service Components to consume and expose through necessary functions of this particular application assets. For example, both legacy applications have their own dedicated adaptors. “Adapter” is owned by the same organization that owns the application asset system and is legislated to be the only way to access this application asset. Such an adapter is provided as an API to the legacy system.

In order to construct a SOAP-enabled Service Component, a “service enabler” is a typical instrument, as shown in Figure 22. Each Service Component contains a service enabler component if it contains access to legacy systems through adapters. Service Component 1 composes two legacy systems through dedicated adapters and enables the composed service to SOAP access through a Service Enabler 1. Service Component 2 composes two legacy systems through dedicated adapters and enables the composed service to SOAP access through a Service Enabler 2. Note that Application 2 only has one adapter, which is re-used in both Service Component 1 and Service Component 2. There are many ways to deploy and reuse the adapter, as a shared library, a common asset, or as a service itself.

10 Service Layer

10.1 Overview

10.1.1 Summary



(From 7.5.4) The Service Layer consists of the logical representation for all the services. The Service Layer can be thought of as containing the service descriptions for business capabilities, services and IT manifestation used and created during design time, as well as runtime service contracts and descriptions that are used at runtime.

The Services Layer is one of the horizontal layers which provides the business functionality supported in the SOA and describes functional capabilities of the services in the SOA.

The description provides consumers with information needed to invoke the business functions exposed by a provider of the service; ideally, this may be done in a platform-independent manner. Service descriptions may include the following documents or references/links to them:

— descriptions of the abstract functionality offered by the service similar to the abstract stage of a WSDL description (see Reference [14]) (note that the use of WSDL is illustrative and the description can be done in any language supporting description of the functionality);

— policy documents;

— SOA management descriptions;

— attachments that categorize or show service dependencies.

Some of the services in the Service Layer may be versions of other services in the set implying that a significant successor/predecessor relationship exists between them.

This layer contains the contracts which include service descriptions that bind the provider and consumer. Services are offered by service providers and are consumed by service consumers (service requestors). Service Components or existing enterprise applications (such as legacy systems and packaged applications) are responsible for the actual implementation or realization of a service. The Operational and IT Systems Layer supports the runtime environment; therefore, implementation of the service components may reside in or use a container and other ABBs in the Operational and IT Systems Layer.

The Services Layer supports the following:

— functional capabilities or services that enable business capabilities those businesses perform in order to achieve a business outcome;

— supporting capabilities to define and specify the “services” in terms of service description;

— supporting capabilities to enable the runtime execution of service and the support of service virtualization.

10.1.2 Context and Typical Flow

The Services Layer introduces the notion of services which are well-defined interfaces for a capability into the architecture with the advent of SOA.

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

Service dependencies can capture the relationships between services where one service is using another, as well as dependencies services have on infrastructure and technologies. Typically, the relationships between services within a service composition are not advertised but rather encapsulated in the composition. Similarly, relationships between services for a business process are captured in a process description. The authoritative source of information on the various versions of a service should be sought from the Governance Aspect which houses and centralizes the service registry/repository.

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

— to identify and define services;

— to provide a container which houses the services;

— to enable use of a registry/repository that virtualizes runtime service access;

— the enable use of a registry/repository to house and maintain service design-time information.

10.1.3 Capabilities

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

— **Service Definition**: This category of capabilities provides the ability to define the service description.

— **Service Runtime Enablement**: This category of capabilities provides the ability to support service versioning, to support service binding decoupling a service from its implementation, and provides the ability to provision services.

— **Policy Management**: This category of capabilities provides the ability to manage and enforce policies associated with services.

— **Access Control**: This category of capabilities provides the ability to manage access to services.

— **Service Clustering**: This category of capabilities provides the ability to cluster services.

This layer features the following supported capabilities.

— **Service Definition**

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

— **Service Runtime Enablement**

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

3) Ability to enable the service container and the service registry/repository 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 aspect

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

6) Ability to support the hosting of services

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

— **Policy Management**

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

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

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

11) Ability to support the audit and logging of runtime service usage to support QoS attributes, with the potential use of standards such as CBE and XDAS to ensure consistent and interoperable data which can then be easily integrated with the Management and Security Aspect 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 Management and Security Aspect of the SOA RA

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

— **Service Clustering**

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

15) Ability to distribute services which are contained by the service provider to invoke layers such as the integration aspect

10.1.4 Structural Overview of the Layer

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

The service runtime environment needs to

— provide runtime support for services,

— provide the container to support runtime service lifecycle management,

— separate out service types and versions and invoke these services,

— support scalability, which becomes critical with high-volume service invocations,

— integrate cross-cutting Aspects, allowing access control, audit and identity integration (security policies), and QoS policies to be integrated, and

— support the actual conversion and binding to the platform for an individual service.

It is important to note that the ABBs in the Services Layer enable both design-time capabilities and runtime capabilities. For example, the Service ABB and Policy Manager supports design-time capabilities and the Service Container ABB supports runtime capabilities. Some ABBs like the Service Registry/Repository in the Governance Aspect ABB support both.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 10.1.3 that the ABB supports. For example, in Figure 23, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Governance Aspect is shown as lighter grey (with a prefix of "Governance"). For example, in Figure 23, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. Therefore, MaS: Policy Enforcer supports capability number 10 and 11: Ability to enforce policies and access control during service binding and Governance: Service Registry/Repository supports capability 1.

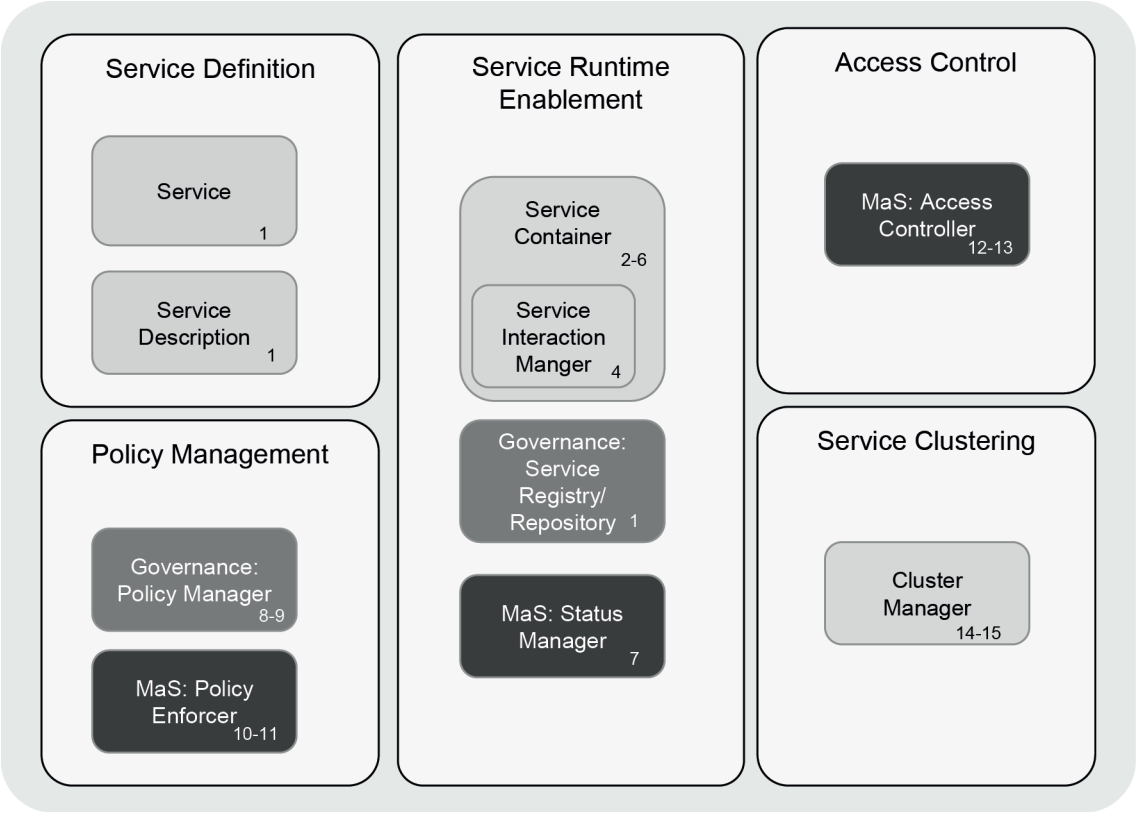


Figure 23 — ABBs in the Services Layer

Figure 23 illustrates the ABBs in the Services Layer and ABBs from other layers that are core to fulfilling the responsibilities of the Services Layer.

ABBs supporting design-time needs are

— Service ABB,

— Service Registry/Repository ABB in the Governance Aspect, and

— Policy Manager ABB in the Governance Aspect.

ABBs supporting runtime environment for the services are

— Service Container ABB,

— Service Interaction Manager ABB,

— Service Registry/Repository ABB in the Governance Aspect,

— Policy Enforcer ABB in the Governance Aspect,

— Access Controller ABB in the Management and Security Aspect,

— Cluster Manager ABB, and

— Status Manager ABB in the Management and Security Aspect.

The details of the ABBs in 10.2 are grouped by the capabilities.

10.2 Details of ABBs and Supported Capabilities

10.2.1 Service Definition

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

10.2.1.1 Service

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

10.2.1.2 Service Description

This ABB represents a service description that contains the information necessary to interact with the service and generally includes in such terms as the service interface (service inputs, outputs, and associated semantics) and service policies (conditions for using the service). Service contracts may reference information in service descriptions.

The service description allows prospective service consumers to evaluate if the service is suitable for their current needs and establishes whether a service consumer satisfies any requirements of the service provider.

10.2.2 Service Runtime Enablement

10.2.2.1 Service Container

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

Thus, its key communication and virtualization responsibilities include the invocation and execution of services, encapsulating the components that implement the service (i.e. providing the service endpoints), state management, and the binding of service invocations to cross-cutting Aspects (such as the Integration Aspect and the 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 and IT Systems Layer. For instance, a Service Container may be contained within a Java EE environment or a .NET environment. It is also possible for it to be a hardware device as long as it provides the ABBs required with the ability to support runtime invocation and running of services and integration with cross-cutting Aspects.

Within the Service Container, there are ABBs which enable it to invoke and execute service components and support the integration with the cross-cutting Aspects, the Management and Security Aspect, Integration Aspect, and Governance Aspect. It leverages the Service Registry/Repository ABB in the Governance Aspect to support service versioning and virtualization.

10.2.2.2 Governance Aspect: Service Registry/Repository

See 16.2.2.3.

10.2.2.3 Service Interaction Manager

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

10.2.2.4 Management and Security Aspect: Status Manager

See 14.2.4.3.

10.2.3 Policy Management

10.2.3.1 Governance Aspect: Policy Manager

See 16.2.4.4.

10.2.3.2 Management and Security Aspect: Policy Enforcer

See 14.2.7.1.

10.2.3.3 Access Control

10.2.3.4 Management and Security Aspect: Access Controller

See 12.2.2.8.

10.2.3.5 Service Clustering

10.2.3.6 Cluster Manager

This ABB represents capabilities to enable scalability within the Services Layer. It provides clustering and caching support when necessary.

10.3 Inter-Relationships between the ABBs

Figure 24 shows the inter-dependencies between the ABBs during design time and runtime.

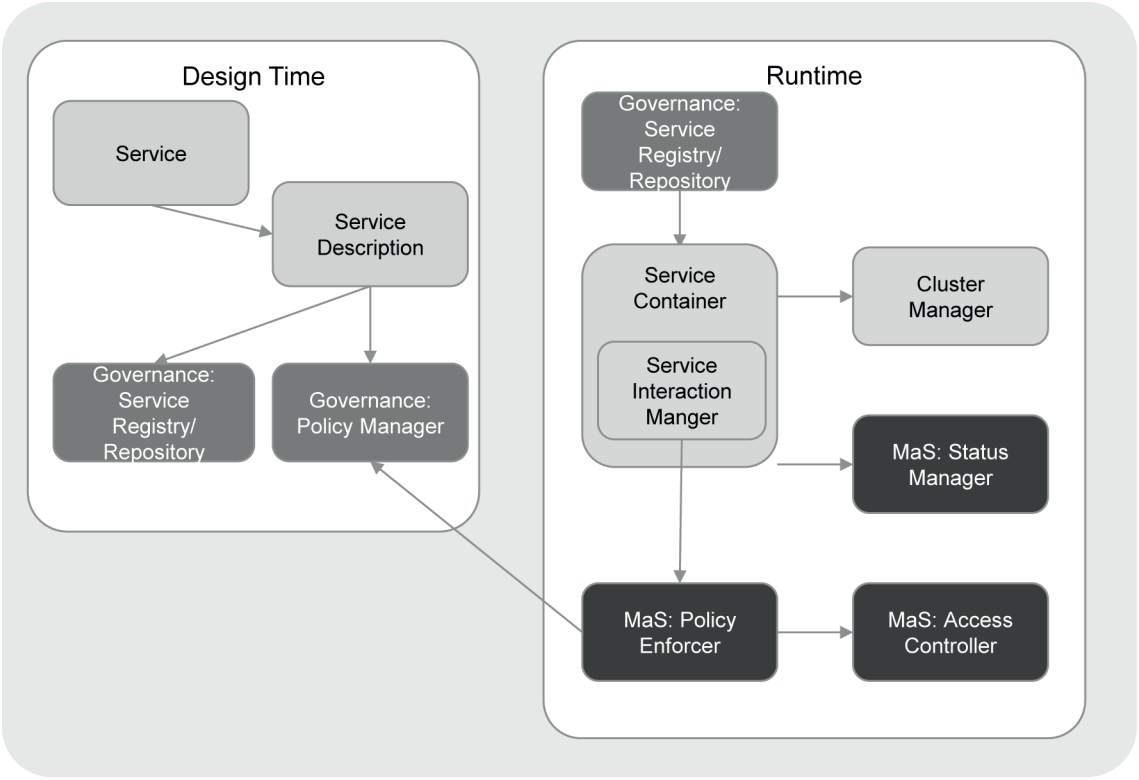


Figure 24 — Relationships among ABBs in the Services Layer

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

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

Usage of a service by a consumer involves two steps: service discovery and location (see Figure 25) and service invocation (see Figure 26). During discovery, in step (1), the service consumer sends a request to the Service Registry/Repository ask for it search or find a service according to some interface or metadata. In step (2), the Service Registry/Repository finds the service endpoint that matches the criteria and returns the endpoint and service description back to the service consumer. Figure 25 shows these steps.

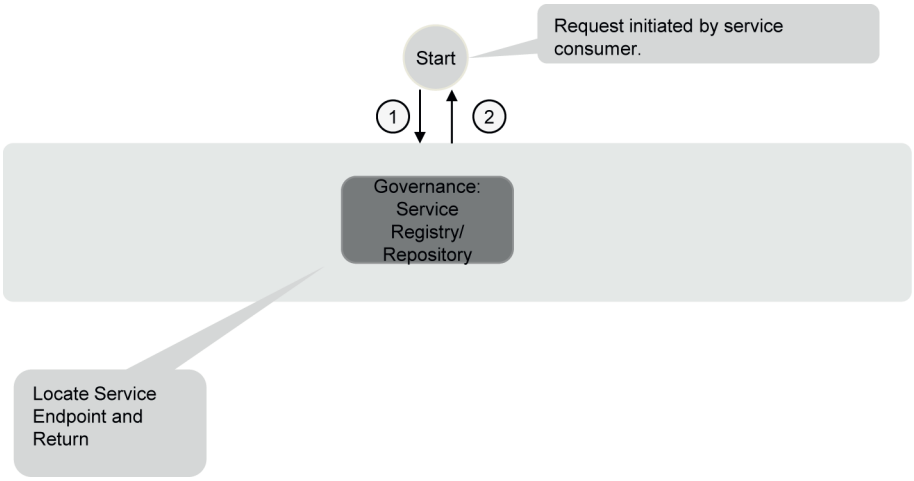


Figure 25 — Interaction Flow for Service Discovery and Location

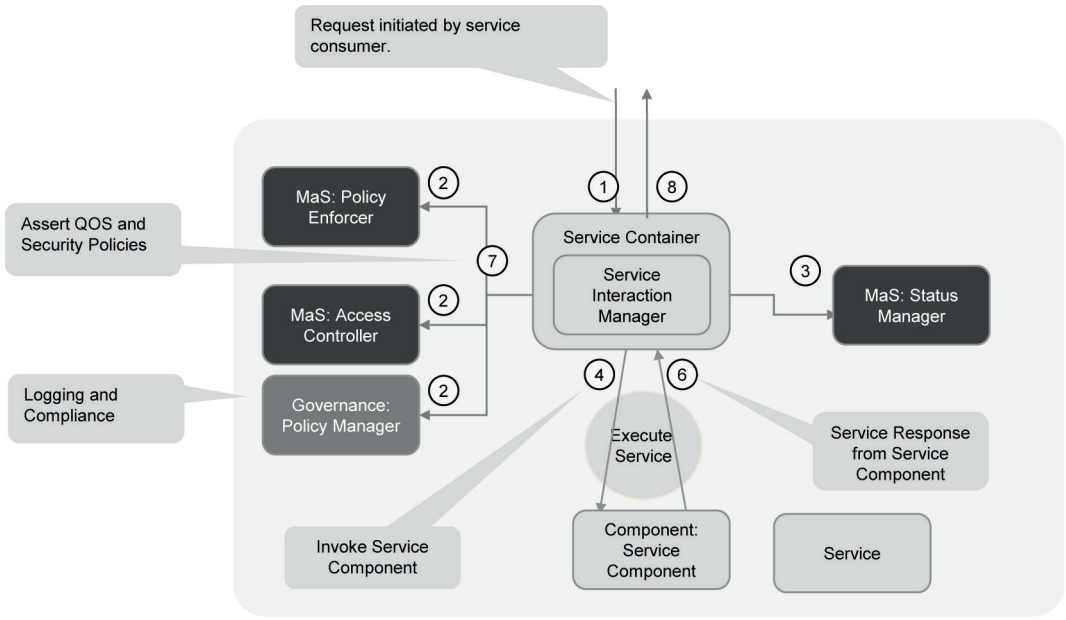


Figure 26 — Interaction Flow for Service Invocation

The second step for the consumer is service invocation as Illustrated in Figure 26. In this case, the consumer initiates an interaction and the request (1) goes to the Service Container for the Service Endpoint, which hands it to the Service Interaction Manager to coordinate the response and all the QoS to be applied to the invocation. First in (2) the Service Interaction Manager invokes the Policy Enforcer to ensure QoS and security policies are complied with. The Access Controller is invoked with the policies and credential from the Policy Enforcer to check to see if the consumer is allowed access to the service. Now, the Service Interaction Manager checks the Policy Manager to see what logging is required for the service for the policy enforcement and invocation of the service. Now that it is sure that the consumer is allowed access, the Service Interaction Manager (3) checks with the Status Manager to see if the service is available. If it is, it executes the services by (4) invoking the appropriate Service Component. When the service execution is finished, it (5) returns the service response to the Service Interaction Manager. The service response (6) is checked against policies and logging is done. When it passes, the Service Interaction Manager hands the service response to the Service Container (7) who returns the response to the consumer.

10.4 Significant Intersection Points with other Layers

10.4.1 Interaction with Cross-Cutting Aspects

The Services Layer relies on cross-cutting aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

It relies on the Development Aspect for the following capabilities:

— ability to implement and test Service with tools;

— ability to create service descriptions, contracts, and deployment descriptions which can be used to advertise and access the service in the service container;

— ability to provision services.

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

The Policy Manager ABB in the Governance Aspect, Access Controller ABB in the Management and Security Aspect, and Policy Enforcer ABB in the Management and Security Aspect exchange and enforce policies ensuring standards-compliant interaction that adheres to the governance regimen. Figure 27 illustrates these relationships.

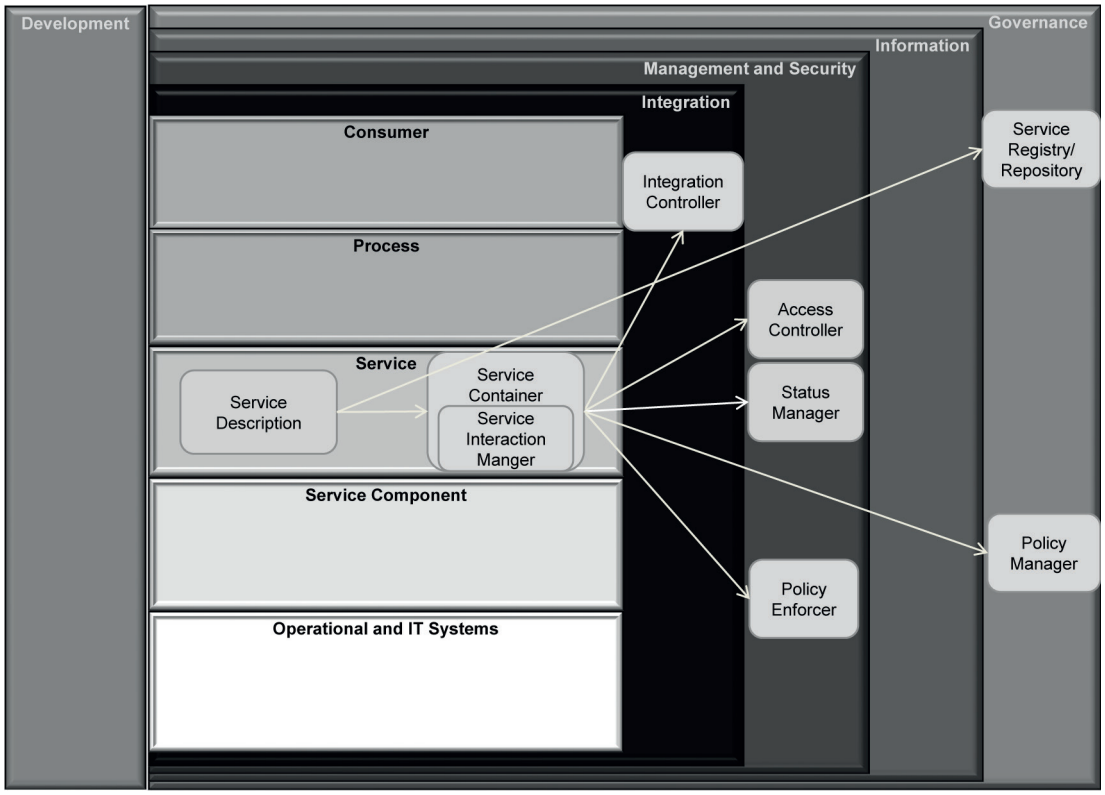


Figure 27 — Interactions from the Services Layer to the Cross-Cutting Aspects

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

10.4.2 Interaction with Horizontal Layers

The Service Registry/Repository ABB in the Governance Aspect 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 Management and Security Aspect to assert the QoS contract of the service and to invoke the Service Interaction Manager to convert invocations into Service Component invocations thus effectively executing the associated service functionality. Finally, Service Components upon completion of the service execution send back the service response data to the Service Interaction Manager which then propagates the service response data back to the service consumer. During execution, if the status of the service changes, the Service Interaction Manager notifies the Status Manager in the Management and Security Aspect of the change. Likewise, the Status Manager can interact with the Service Interaction Manage to change the status of the service.

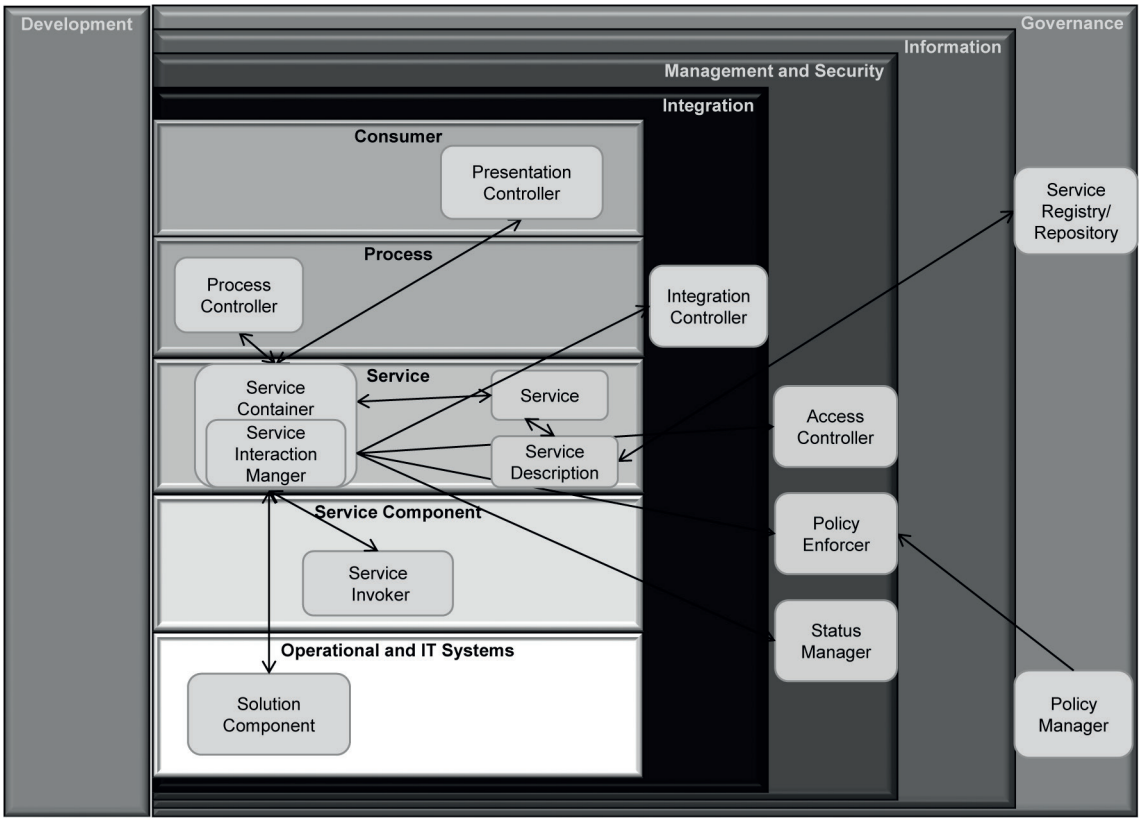


Figure 28 — Interaction with Horizontal Layers

To summarize:

— The Service Registry/Repository ABB in the Governance Aspect provides design-time storage of metadata for services;

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

— The Service Interaction Manager ABB plays three roles: acting as the outward-facing ABB which is invoked by other layers at runtime to invoke services, invoking the service components from the service components layer, and coordinating conversion of data between different formats;

— The service binder invokes the service container to compose all the information required to invoke the service ABB. The Service Interaction Manager ABB uses the Policy Enforcer ABB and Access Controller ABB in the Management and Security Aspect to enforce and incorporate any security and QoS policies. It uses the state manager to address any state-related issues. It then calls the Service Invoker ABB in the Service Component Layer to execute the service functionality, accept the result from the service component, interact with the Service Interaction Manager ABB, and propagate it back via the Service Container ABB to the invoking consumer;

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

— The Policy Enforcer ABB provides the interaction point, between the Management and Security Aspect and the Service Layer, supporting Policy Enforcement. The Access Manager provides Authorization and Authentication support in the context of the layer and integrates with corresponding ABBs which define security policies in the Management and Security Aspect. In practice, it too acts as a policy enforcer.

10.5 Usage Implications and Guidance

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

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

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

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

Services are accessible independent of the implementation and transport. This allows a service to be exposed consistently across multiple customer-facing channels such as the web, Interactive Voice Response (IVR), etc. The transformation of response to HTML (for the web), VoiceXML (see Reference [26]) (for IVR), and XML string (for XML client) can be done via XSLT (see Reference [25]) functionality supported through transformation capability in the integration aspect.

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.

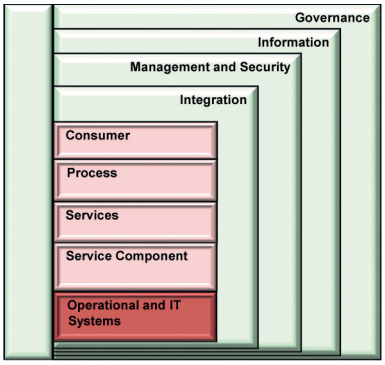
In addition to being an important template for defining an SOA solution at a logical level, the SOA RA is also a useful tool in the design of vendor-neutral SOA solutions because it enables the objective identification of SOA infrastructure requirements that can be used to leverage the various capabilities provided by a mix of different vendors who may offer the same ABB.

Using the same SOA RA, SOA business services can be delivered based on the same deployment framework.

11 Process Layer

11.1 Overview

11.1.1 Summary



(From 7.5.5) The Process Layer covers the process representation, composition methods, and building blocks for aggregating loosely coupled services as a sequence of steps aligned with business goals. Data flow and control flow are used to enable interactions between any combination of services and business processes. The interaction may exist within an enterprise or across multiple enterprises.

Business capabilities are realized through the execution of one or more business processes. These business processes may be realized through service compositions (e.g. orchestrations, choreographies, or collaborations) and include support for manual human interactions and long-lived transactions. The evolution of service composition into flows (e.g. choreographies of services bundled into a flow) can act together to establish an SOA solution. These SOA solutions support specific use cases and business processes.

This layer includes information exchange flows between participants (individual users and business entities), resources, and processes where the exchanged information may include unstructured and non-transactional messages. The business logic is used to form service flows as parallel tasks or sequential tasks based on business rules, policies, and other business requirements.

The Process Layer performs three-dimensional process-level handling: top-down, bottom-up, and horizontal. From the top-down direction, this layer provides capabilities and ABBs to help 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 service-oriented collaboration control between business processes, services, and service components.

In summary, the Process Layer in the SOA Reference Architecture plays a central coordinating role in connecting business level requirements and IT-level solution components through collaboration with the Integration Aspect, Management and Security Aspect, Information Aspect, Governance Aspect, and Services Layer.

11.1.2 Context and Typical Flow

The Process Layer allows externalization of the business process flows in the architecture and supports the ability to change business processes, processes and flows as the market condition changes. When processes are embedded in software components and user interfaces, they can be difficult document and change. Using the Process Layer to explicitly represent processes, compositions, and collaborations in the architecture makes them easier to document, maintain and change.

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 solution or, at best, a hardwired workflow. With service orientation, a process can be realized by service compositions (which may be interim as an orchestration or a choreography) and the ability to insert "human intervention" and support long-running transactions.

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

Services combined or composed into flows or, for example, choreographies of services, bundled into a flow, work together to establish a solution. These solutions support specific use-cases and business processes. Typically, a visual flow composition tool will be used to design the solution flow. Figure 29 shows how a Business Process "P" can be implemented using Services A, B, C, and D from the Services Layer. Process P contains the logic for the sequence in which the services need to be invoked and executed, as well as having support for many of the non-functional concerns such as state management. The services that are aggregated into a business process can be sourced from individual services or composite services.

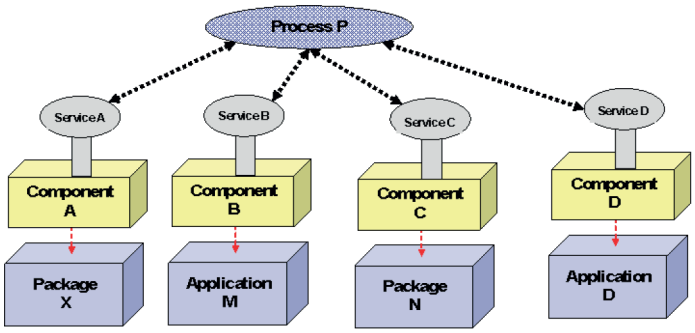


Figure 29 — Services Orchestration

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

Understanding Business Processes

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. Building process blocks on-demand with reduced cost allows the supporting technology to change from being high volume/transactional to sophisticated but much smaller footprint solutions.

The Process Layer supports three-dimensional process-level handling: top-down, bottom-up, and horizontal.

— From the top-down approach, a business process can be defined by business people based on customers' requirements. In order to optimize the business process for better IT implementation, a business process should be componentized as re-usable services that can be modelled, analysed, and optimized based on business requirements such as Quality of Service (QoS) (historical data described in the Management and Security Aspect), flow preference, price, time of delivery, and customer preferences. From the top-down direction, this layer supports facilities to decompose business requirements into tasks comprising activity flows, each being realized by existing business processes, services, and service components.

— The decomposition of a business process first decomposes it into smaller tasks, then each task is mapped into coarse-grain service (i.e. candidate services) that will be realized by actual web

services in the Services Layer. In other words, the layer provides the ability to decompose a business process into coarse-grain candidate services that fulfil the business functions.

— From the bottom-up approach, the layer provides facilities to compose existing business processes, services, and service components into new business processes. 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 by reusing services to address different types of customer pinpoints.

— From the horizontal approach, the layer provides services-oriented collaboration control between business processes, services, and service components. From an interaction perspective, the 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 Aspect. The contents of the messages could be defined by the Information Aspect. The Key Performance Indicators (KPIs) for each task or process could be defined in the Management and Security Aspect. The aggregation of services could be guided by the Governance Aspect.

All the services should be represented and described by the Services Layer and Service Components are represented by the Service Component Layer.

The following are specific considerations in applying processes to web services.

— Since business processes are driven by business requirements, which typically tend to be informal, subjective, and difficult to quantify, 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.

— 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.

— Present web services description generally lacks a facility to define comprehensive relationships among business entities, business services, and operations. These relationships may be important to optimize business process composition. For example, suppose enterprise E1 needs to compose a business process including service S. Enterprises E2 and E3 both provide similar service S. However, there is a partnership between E1 and E2 that leads to a discount on services and there is no partnership relationship between E1 and E3. If price is a requirement for party E1, this partnership between E1 and E2 needs to be taken into consideration in order to form the most appropriate business process.

— Since so many web services are published to the Internet on the daily basis, it can be a challenge to clearly specify search requirements to discover the most appropriate web services candidates.

— Business processes typically requires multiple web services to collaborate in order to serve business requirements.

All of these issues mean that each Service Component not only needs to satisfy individual requirements but also needs to coexist with other Service Components in order to best fit the overall composed business process. In other words, the entire business process needs to be optimized prior to execution.

The capabilities of the Process Layer enable solutions to address those challenging issues because the capabilities and ABBs in the Process Layer play a central coordinating role in connecting business-level requirements and IT-level solution components of the SOA solution.

One way SOA can support business processes of varying granularity is to use the Process Flow Manager, which manages one or more process instances (representation of a single running business process) concurrently. The ABB has a core responsibility to split and manage processes and its sub-processes together and enabling services. The Process Flow Manager can support multiple interfaces that interact

with business processes, i.e. both an SCA interface for interacting with other service components and a generic process API for interacting with Java EE clients. The Process Flow Manager works with the Workflow Manager or Human Task Manager ABB.

For example, a Process Claim business process might have underlying business processes that support the Process Claim business process. This ABB, in this case, then manages all the interactions and the decomposition relationships for that individual's sub-processes and enabling services.

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.

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

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

11.1.3 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 Process Layer needs to support. These categories of capabilities are as follows.

— **Process Definition**: This category of capabilities is required for defining the business processes/operational flow of the business.

— **Event Handling**: This category of capabilities handles business events in the context of a business process such as emitting/publishing events and subscribing/listening to business events.

— **Process Runtime Enablement**: This category of capabilities enables BPM and helps to realize the business processes in the runtime environment using standards such as BPEL, SCA, etc.

— **Process Information Management**: This category of capabilities manages the information needs of a business process such as managing its state, transforming data in the process flow, and maintaining a registry/repository of assets.

— **Decision Management**: This category of capabilities defines and manages the decision points and associated rules within a business process.

— **Process Integration**: This category of capabilities facilitates integration with others layer of the SOA RA and helps to expose a business process as a service.

— **Security and Policy Compliance**: This category of capabilities enables access control and policy enforcement in the business processes.

— **Process Monitoring and Management**: This category of capabilities monitors and manages business processes, identifies bottlenecks in the business processes, and optimizes workload assignment.

This layer features the following capabilities.

— **Process Definition**

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

— **Event Handling**

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

— **Process Runtime Enablement**

3) Ability to realize and deploy the business processes in a runtime environment

4) Ability to create and manage individual instances of the business processes

5) Ability to execute the instances of a business process, its sub-processes, and activities therein

6) Ability to define the elements of an assembly at design time and have the assembly occur at runtime based on a set of rules

7) Ability to intelligently decide the endpoint of the enabling services using the process context

8) Ability to manage the interaction of the business process with humans

— **Process Information Management**

9) Ability to manage the context of a business process

10) Ability to manage the state of a process

11) Ability to transform the data flowing through a business processes based on its needs

12) Ability to store and retrieve assets required and requested by an ongoing process

— **Decision Management**

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

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

— **Process Integration**

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

16) Ability to schedule the execution of a business process

— **Security and Policy Compliance**

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

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

— **Process Monitoring and Management**

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

11.1.4 Structural Overview of the Layer

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

— definition, composition, and decomposition of a business process,

— handling of events,

— runtime enablement and realization of business processes,

— management of information and associated data flow in the context of the business processes,

— management of decision points/points of variability and associated rules in the context of the business processes,

— integration capabilities necessary for realizing business processes,

— security and policy compliance pertaining to business processes, and

— monitoring and management of business processes.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 11.1.3 that the ABB supports. For example, in Figure 30, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:'), while the ABB from the Integration Aspect is shown as black (with a prefix of "Integration"). For example, in Figure 30, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. MaS: Policy Enforcer supports capability number '17: Ability to define policies, enforce them, and verify the compliance of the elements of process with a set of predefined policies'. The Integration: Data Transformer supports capability '11: Ability to transform the data flowing through a business processes based on its needs'.

Figure 30 illustrates the ABBs partitioned into key categories.

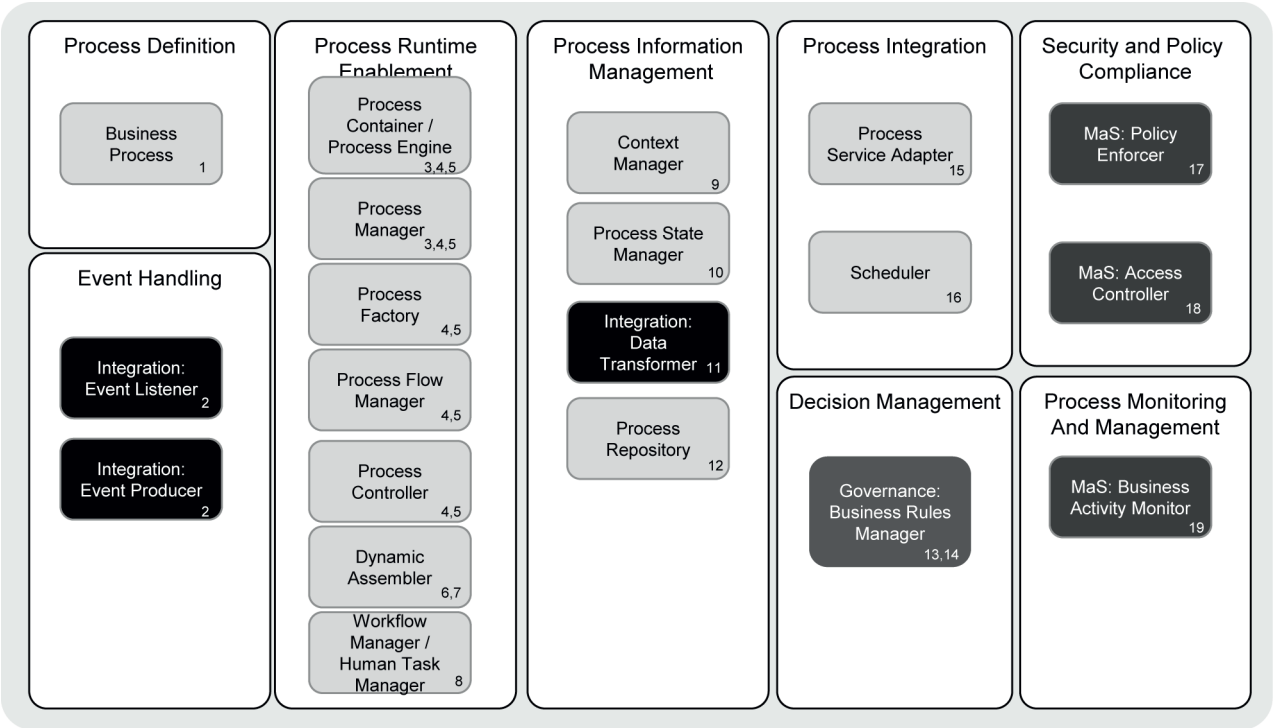


Figure 30 — ABBs in the Process Layer

The details of the ABBs in 11.2 are grouped by the capabilities This Clause describes each of the ABBs in the Process Layer in terms of their responsibilities.

11.2 Details of ABBs and Supported Capabilities

11.2.1 Process Definition

11.2.1.1 Business Process

This ABB represents a process that a business performs, including the implementation of the process, business logic, sources of inputs, and outputs. This ABB is one of the core fundamental ABBs in the SOA RA. Business process is one of the fundamental constructs of an SOA solution and analysis and design based on the service oriented paradigm.

11.2.2 Event Handling

11.2.2.1 Integration Aspect: Event Listener

See 13.2.2.7.

11.2.2.2 Integration Aspect: Event Producer

See 13.2.2.6.

11.2.3 Process Runtime Enablement

11.2.3.1 Process Container/Process Engine

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

11.2.3.2 Process Manager

This ABB represents capabilities for deploying processes in the process container.

11.2.3.3 Process Factory

This ABB represents capabilities for creating instances of deployed processes in the process container.

11.2.3.4 Process Flow Manager

The ABB represents capabilities for managing process templates and process instances. Process templates describe the business process model. At runtime, this ABB creates process instances (representation of a single running business process) from process templates using the Process Factory ABB and manages one or more process instances concurrently.

11.2.3.5 Process Controller

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

11.2.3.6 Dynamic Assembler

This ABB represents capabilities 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 decisions on which endpoint is to be invoked.

11.2.3.7 Workflow Manager/Human Task Manager

This ABB represents capabilities 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 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).

11.2.4 Process Information Management

11.2.4.1 Context Manager

The ABB represents capabilities for managing the context of various instances of the business process.

11.2.4.2 Process State Manager

The ABB represents capabilities that 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.

11.2.4.3 Integration Aspect: Data Transformer

See 13.2.2.4.

11.2.4.4 Process Repository

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

11.2.5 Process Integration

11.2.5.1 Process Service Adapter

This ABB represents capabilities for integrating the Process Layer with other SOA RA layers, in particular, the Services, Integration, and Consumer Layers. It is a type of adapter that acts as the mechanism to expose business processes as services.

11.2.5.2 Scheduler

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

11.2.6 Decision Management

11.2.6.1 Governance Aspect: Business Rules Manager

See 16.2.3.2.

11.2.6.2 Security and Policy Compliance

11.2.6.3 Management and Security Aspect: Policy Enforcer

See 14.2.7.1.

11.2.6.4 Management and Security Aspect: Access Controller

See 14.2.2.8.

11.2.7 Process Monitoring and Management

11.2.7.1 Management and Security Aspect: Business Activity Monitor

See 14.2.5.2.

11.3 Inter-Relationships between the ABBs

Figure 32 shows the inter-relationships and a non-normative sequence of interaction. The final, prescriptive sequence of interaction will be defined by the underlying solution architecture and the standards that are invoking the support of other SOA RA layers. The Consumer invokes the Process Service Adapter which calls the Process Container or Process Engine which calls the Process Flow Manager. Process Flow Manager invokes any other appropriate ABBs as indicated in Figure 31. After the Process completes, the Process Flow Manager calls the Process Controller which uses the Data Transformer from the Integration Aspect before it returns the results to the Provider through the Process Service Adapter.

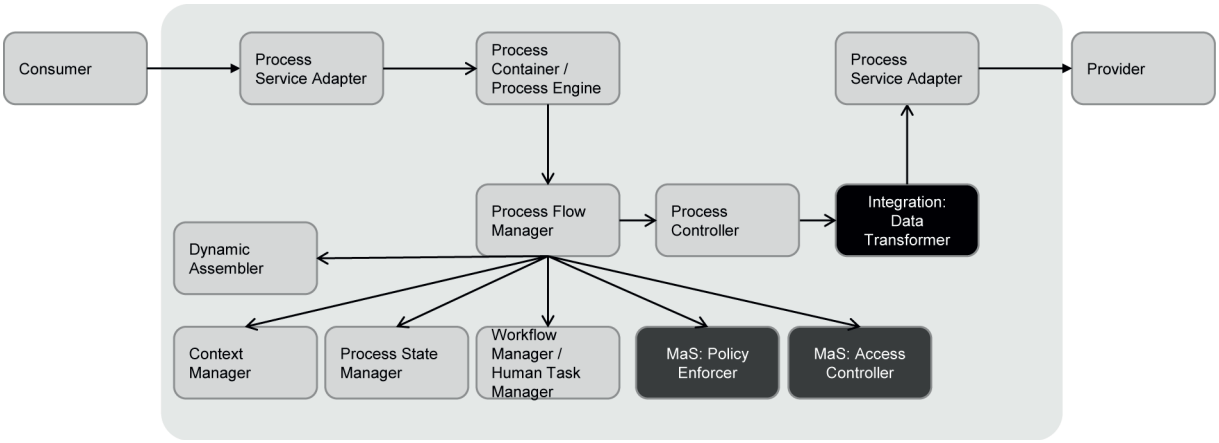


Figure 31 — Key Relationships among ABBs in the Process Layer

11.4 Significant Intersection Points with other Layers

11.4.1 Interaction with Cross-Cutting Aspects

The Process Layer relies on cross-cutting aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

It relies on the Development Aspect for the following capabilities:

— ability to design, simulate, and optimize business processes;

— ability to develop and change business processes that orchestrate activities and underlying services.

It relies on the Governance Aspect for the following capabilities:

— ability to store metadata for policies;

— ability to support management (storage, retrieval, etc.) of rules to support rules associated with decision points in service mediation, orchestration, and composition.

It relies on the Management and Security Aspect for the following capabilities:

— ability to authenticate/authorize for service invocation.

It relies on the Information Aspect for the following capabilities:

— ability to store and retrieve metadata and data required for the execution of the business processes.

It relies on the Integration Aspect for the following capabilities:

— ability to invoke services to realize systematic process steps;

— ability to transform data from one format to another.

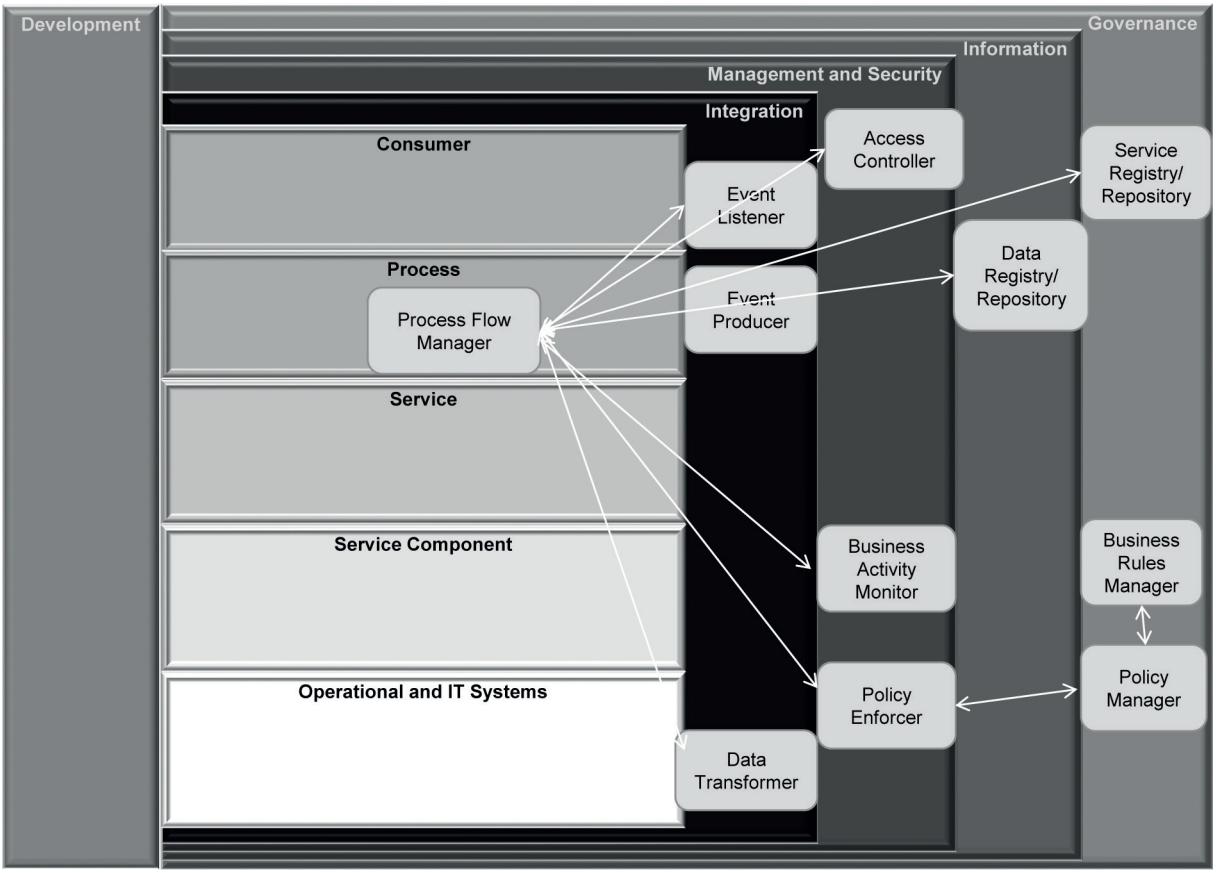


Figure 32 — Key Interactions of the Process Layer with Cross-Cutting Aspects

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

— It leverages the Service Registry/Repository ABB from the Governance Aspect for storing metadata such as policy, schema, etc. and for providing access to the metadata. This Service Registry/Repository 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 endpoint through a "proxy" (the registry/repository).

— It leverages the Business Rule Manager ABB in the Governance Aspect to manage the rules supporting the decision points or points of variability within a business process.

— It leverages the Access Controller ABB in the Management and Security Aspect for authenticating/authorizing the facility for service invocation and workload assignment.

— It leverages Data Registry/Repository ABB from the Information Aspect to store and assess data.

— It leverages the Access Controller ABB in the Management and Security Aspect to enforce access privileges and the Policy Enforcer ABB in the Management and Security Aspect to enforce policies.

— It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Aspect such as data transformation, service request, etc. It leverages the Mediator ABB in the Integration Aspect to integrate with existing systems and applications. It leverages the Data Transformer ABB in the Integration Aspect to transform data from one format to other. It leverages the Event Producer ABB and Event Listener ABB in the Integration Aspect to publish events or subscribe to events.

11.4.2 Interaction with Horizontal Layers

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

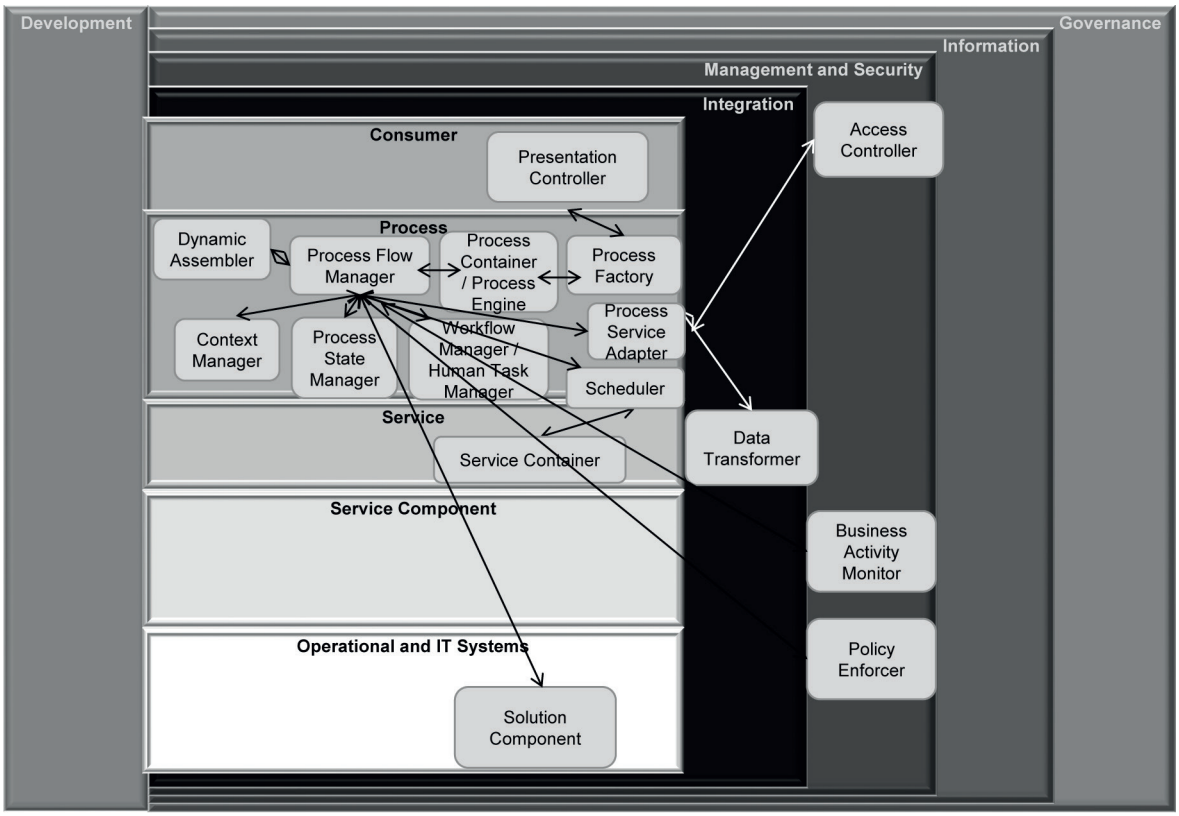


Figure 33 — Key Interactions of the Process Layer with Horizontal Layers

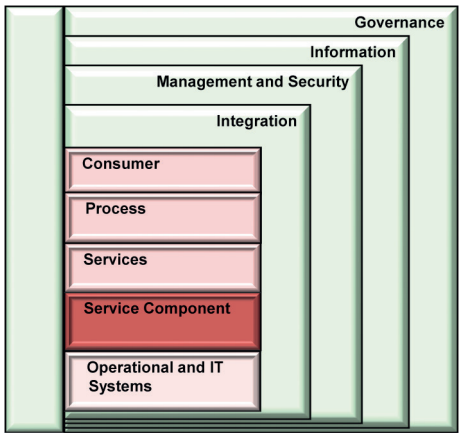
11.5 Usage Implications and Guidance

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

12 Consumer Layer

12.1 Overview

12.1.1 Summary



(From 7.5.6) The Consumer Layer is the point where consumers, either human actors or SOA solutions, interact with the SOA solution or ecosystem. It enables SOA solutions 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). In this discussion, channels can be thought of as the platforms by which SOA consumers access services through the SOA. Examples of channels include front-ends and interactive voice response (IVR) systems, which could both leverage the same core functionality within the SOA. Thus, the Consumer Layer is the point of entry for all internal and external interactive consumers, including services acting as consumer (for example, in B2B scenarios).

For human consumers, the Consumer Layer is often realized through a user interface that accepts requests and returns responses. This interface may enable specific users to customize preferences, integrate with consumer channels, including rich clients such as mashups and Ajax (see Reference [18]) and act as a mechanism for the underlying SOA to expose its functionality. Standards such as Web Services Remote Portlets (WSRP) (see Reference [15]) may leverage services at the application interface or presentation level.

The user interface provides the visible portion of the Consumer Layer capabilities but the Consumer Layer may also incorporate other business processes dictated by policy or desired business outcomes. For example, Consumer Layer capabilities may include the point where in-bound service 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 via collaboration with other Aspects.

For consumers that are other services or SOA solutions, the Consumer Layer points to defined service interfaces, where in parallel with the human use of the Consumer Layer, the service interface may point to a composition which includes the application of other business processes, such as security and quality of service from the Management and Security Aspect. The Consumer Layer provides the capability to quickly create a front end for business processes and other service compositions to respond to changes in business requirements. This “front end” can be a new service interface, a new user interface, or an appropriate combination. It enables channel independent access to those business processes supported by a variety of solutions and platforms.

This decoupling between the consumer and the rest of the underlying SOA provides organizations the ability to support agility, enhance reuse and improve quality and consistency.

12.1.2 Context and Typical Flow

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 Consumer Layer is the point where consumers interact with the SOA.

12.1.3 Capabilities

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

— **Consumer Services**: This category of capabilities addresses the support of interaction with consumers.

— **Presentation Services**: This category of capabilities addresses the support of presentation services, which include a presentation, composite view and presentation control, and the consumer-centric configuration of views.

— **Backend Integration**: This category of capabilities addresses the integration of the Consumer Layer with backend and legacy systems using SOA and services and transforms their information and incorporates it into content.

— **Caching and Streaming Content**: This category of capabilities addresses the support of information buffering and performance and supports the operation of the Consumer Layer.

— **Security and Privacy**: Capabilities that address the support of QoS, information protection, and security.

— **Information Access**: This category of capabilities addresses the sharing of data and metadata across the layers of the SOA RA such as QoS attributes, attributes defining common rules to be used across the layers, etc.

This layer features the following capabilities.

— **Consumer Services**

1) Ability to consume (use) the SOA, through a program or an individual who requests a service

2) Ability to support consumer interaction and integration, i.e. the ability to capture the input from the user (consumer) of the SOA and provide the response to the consumer

— **Presentation Services**

3) Ability to support the creation of a presentation view by the composition of a number of atomic components

4) Ability to configure information which will support specific capabilities associated with ensuring consistency (similar to a style guide)

5) Ability to provide navigation logic and flow for the processing of consumer interactions (presentation control)

6) Ability to provide the Consumer Layer with the ability to support customer-specific information (enabled by the Information Aspect) 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 Process Layer and the Integration Aspect 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 Solution

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

— **Information Access**

15) Ability to access data and metadata through the Information Aspect

12.1.4 Structural Overview of the Layer

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

— ability to support the interaction of the SOA with consumers,

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

— ability to integrate information from services from the SOA and transform their information and incorporate it into content,[3)]

— ability to support information buffering and performance and support the operation of the Consumer Layer,

— ability to support QoS, information protection, and security, and

— ability to address the sharing of metadata across the layers of the SOA RA, and

— ability to support presentation.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 12.1.3 that the ABB supports. For example, in Figure 34, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Integration Aspect is shown as black (with a prefix of "Integration"). For example, in Figure 34, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. MaS: Access Controller supports capability number 12 and 13: '13. ability to filter to control access to the underlying SOA Solution'. The Integration: Data Transformer supports capability ' 9: Ability to support the translation of input data/content from a format supported by the user of the

3) 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.

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'.

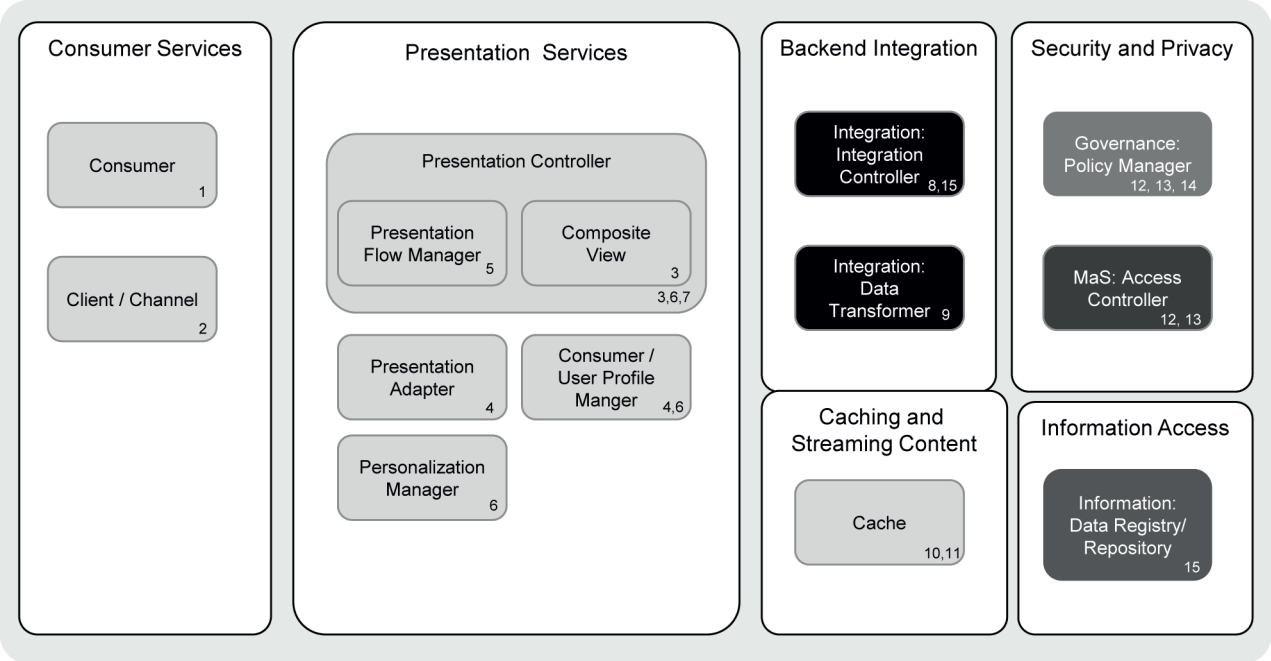


Figure 34 — ABBs in the Consumer Layer

12.2 groups the ABBs by capability and describes the details each of the ABBs in the Consumer Layer in terms of their capabilities.

12.2 Details of ABBs and Supported Capabilities

12.2.1 Consumer Services

12.2.1.1 Consumer

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

12.2.1.2 Client (a.k.a Channel)

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

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.

12.2.2 Presentation Services

12.2.2.1 Presentation Adapter

This ABB represents 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.

12.2.2.2 Presentation Controller

This ABB represents 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.

12.2.2.3 Presentation Flow Manager

This ABB represents 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.

12.2.2.4 Composite View

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

12.2.2.5 Consumer/User Profile Manager

This ABB represents 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.

12.2.2.6 Personalization Manager

This ABB represents the ability of users to customize the appearance of the user interface according to personal preferences. The customization is accomplished partly through administrative set-up, which defines the default settings and access rights to user interfaces/web pages. It supports both rule-based personalization to select content for the user such as a rule might display special discounts to gold customers but only during the summer months.

Collaborative filtering technology-based personalization to select content based on common interests or behaviours.

12.2.3 Backend Integration

12.2.3.1 Integration Aspect: Integration Controller

See 13.2.1.1.

12.2.3.2 Integration Aspect: Data Transformer

See 13.2.2.4.

12.2.4 Caching and Streaming Content

12.2.4.1 Cache

This ABB represents elements used to support the caching of interaction data to improve scalability and performance.

12.2.5 Security and Privacy

12.2.5.1 Governance Aspect: Policy Manager

See 16.2.4.4.

12.2.5.2 Management and Security Aspect: Access Controller

See 14.2.2.8.

12.2.6 Information Access

12.2.6.1 Information Aspect: Data Registry/Repository

See 15.2.7.1.

12.3 Inter-Relationships between the ABBs

In the Consumer Layer, there is a logical partitioning between integration with the consumer, integration with the services components of the SOA, integration with some of the cross-cutting SOA aspects (information and QoS), and the functionality for the creation of service invocations (requests) to the SOA and composing the returned and cached content to the consumer. Consumers can be human or system. The scenarios below illustrate usage of the Consumer Layer. The first scenario shows the interaction with human service consumers initiating the interaction with the SOA. The second scenario shows the interaction with systematic (non-human) service consumers initiating the interaction with the SOA. It is important to note that, practically, there is no real difference between human and non-human actors; they all represent interactions with the SOA. The extent of differences between the two scenarios is really 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 in Figure 35, the Consumer Layer provides the separation of concerns to enable the SOA to have maximum re-usability and agility, leveraging the core services in the Integration Aspect and Process Layer and the capabilities of the rest of the SOA.

In Figure 35, the invocation is being done by a system or service, or any non-human user, as a consumer. The processing and flow should be exactly the same for humans and non-humans and is apparent in Figure 35.

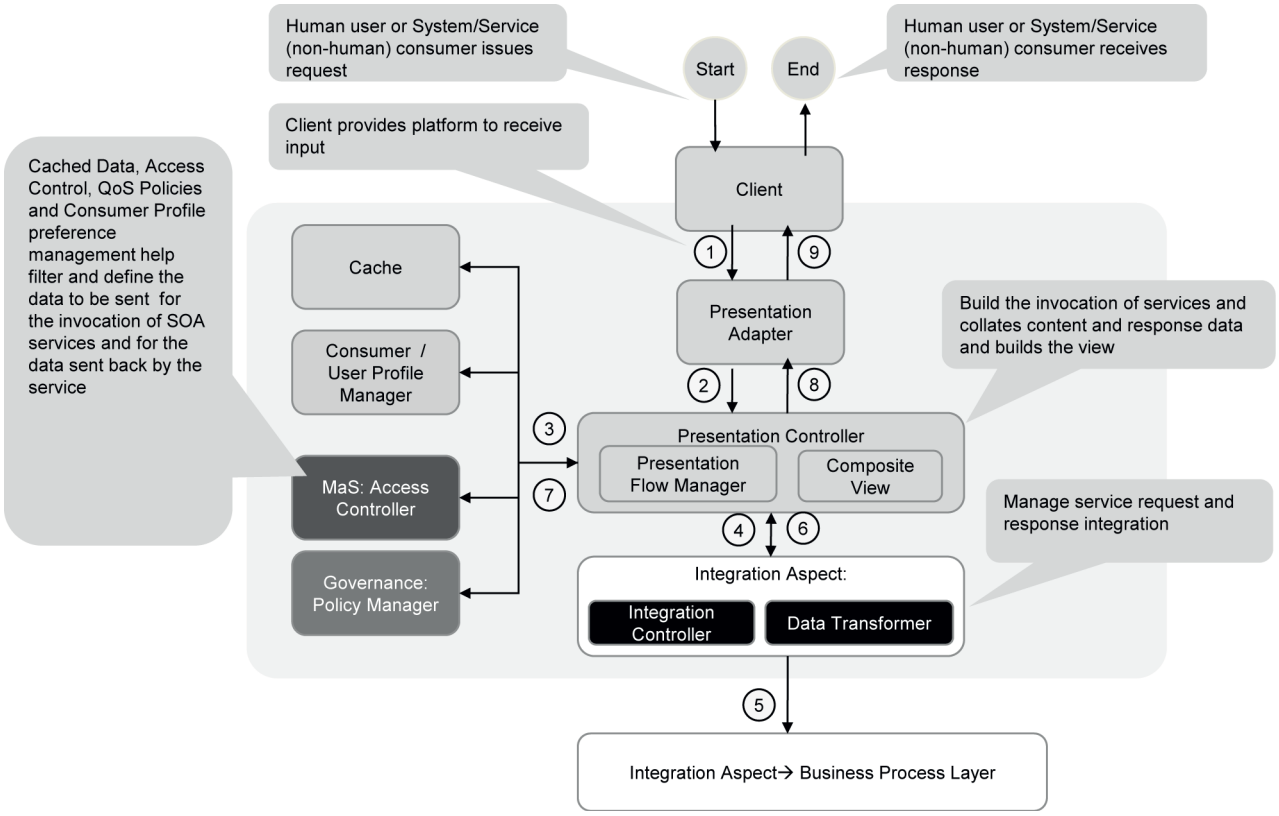


Figure 35 — Interaction of a Systematic (Non-Human) Service Consumer with the SOA via the Consumer Layer

In Figure 36, the service is being used or invoked by another service or system. Note that in a well-designed SOA solution and as illustrated here, the difference in the invocation will be isolated in the client and the rest of the flow for invoking the service will be same as when being invoked by a human user.

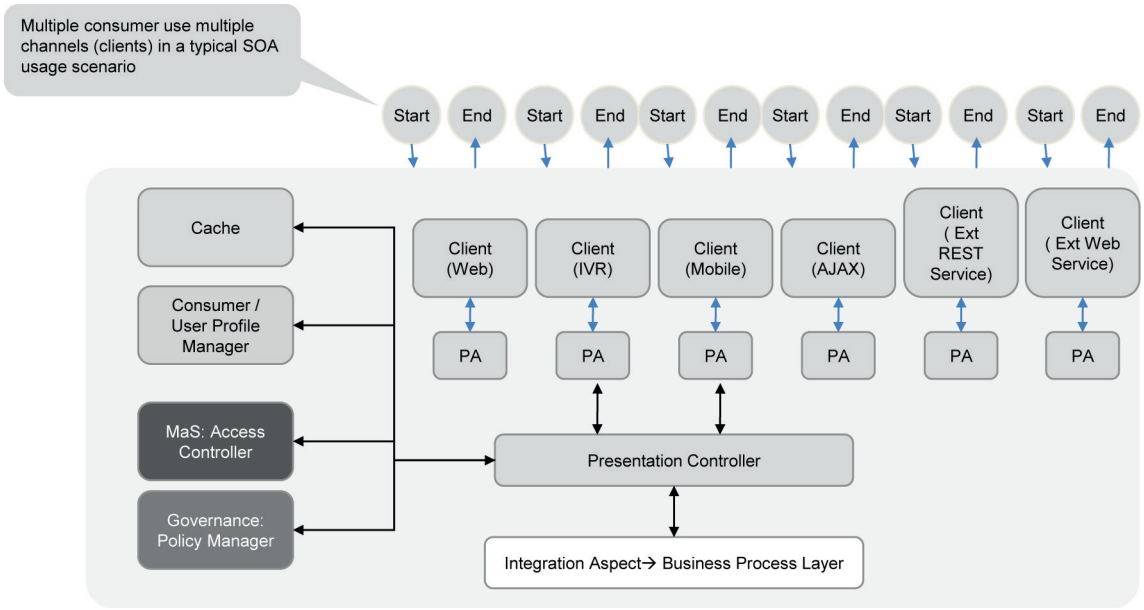


Figure 36 — Typical SOA Usage Scenario with Multiple Consumers using Multiple Channels

Figure 36 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 multiple instances of the ABBs is shown).

12.4 Significant Intersection Points with other Layers

12.4.1 Interaction with Cross-Cutting Aspects

The Consumer Layer relies on cross-cutting aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices. It relies on the Development Aspect for the following capabilities:

— ability to develop and test consumer and channel implementations with tools;

— ability to process service descriptions Development Aspect.

It relies on the Governance Aspect for the following capabilities:

— ability to store metadata for policies.

It relies on the Management and Security Aspect for the following capabilities:

— ability to authenticate/authorize for service invocation.

It relies on the Information Aspect for the following capabilities:

— ability to store and retrieve metadata and data.

It relies on the Integration Aspect for the following capabilities:

— ability to invoke business processes and/or services;

— ability to transform data from one format to another.

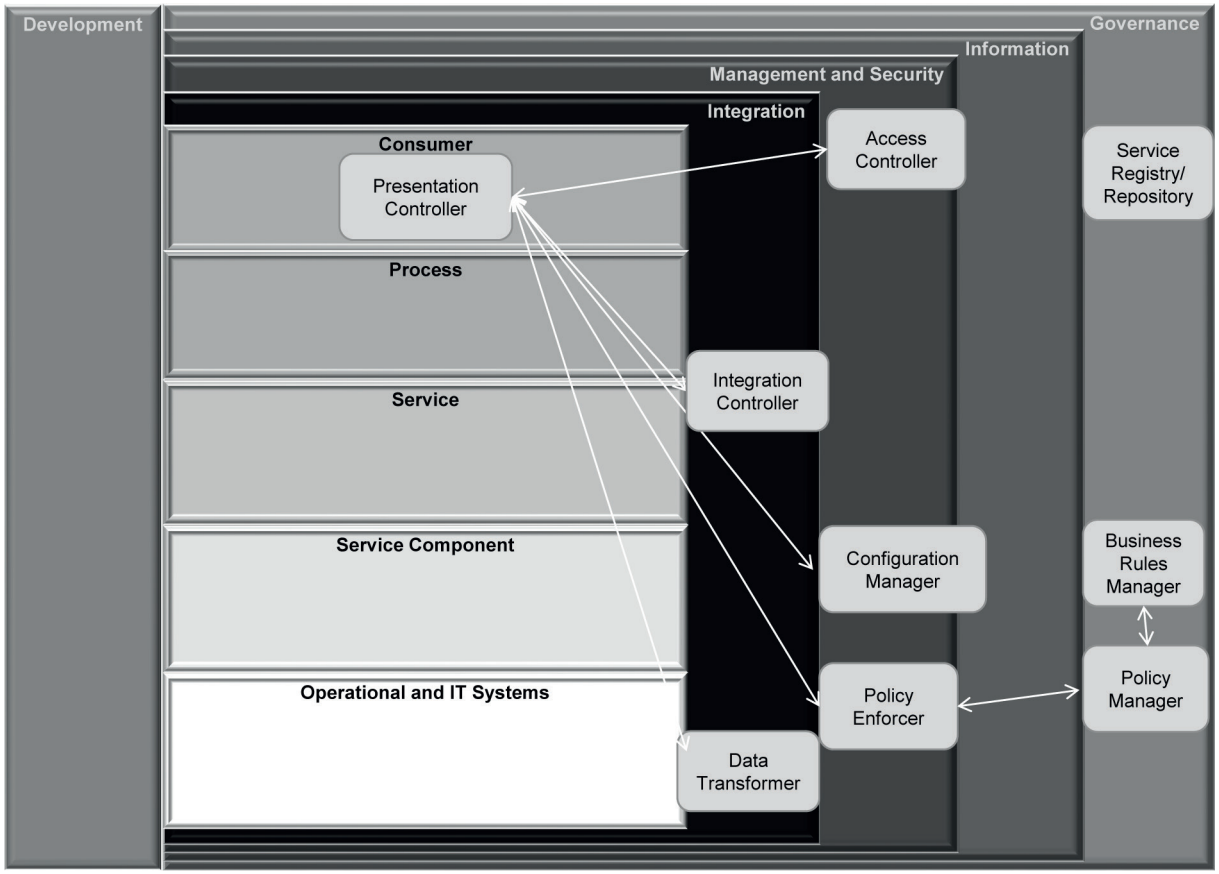


Figure 37 — Key Interactions of the Consumer Layer with Cross-Cutting Aspects

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

— It leverages the Access Controller ABB and Policy Enforcer ABBs in the Management and Security Aspect to enforce access control privileges and other policies.

— It leverages the Data Aggregator ABB, Data Federator ABB, Data Consolidator ABB, Information Metadata Manager ABB, and Data Registry/Repository ABB from the Information Aspect to store and assess data.

— It interfaces with the Integration Controller ABB to leverage the capabilities of the Integration Aspect such as data transformation, service request, etc. It leverages the Mediator ABB in the Integration Aspect to integrate with existing systems and solutions. It leverages the Data Transformer ABB in the Integration Aspect to transform data from one format to other. It leverages the Event Producer ABB and Event Listener ABB in the Integration Aspect to publish events or subscribe to events.

12.4.2 Interaction with Horizontal Layers

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

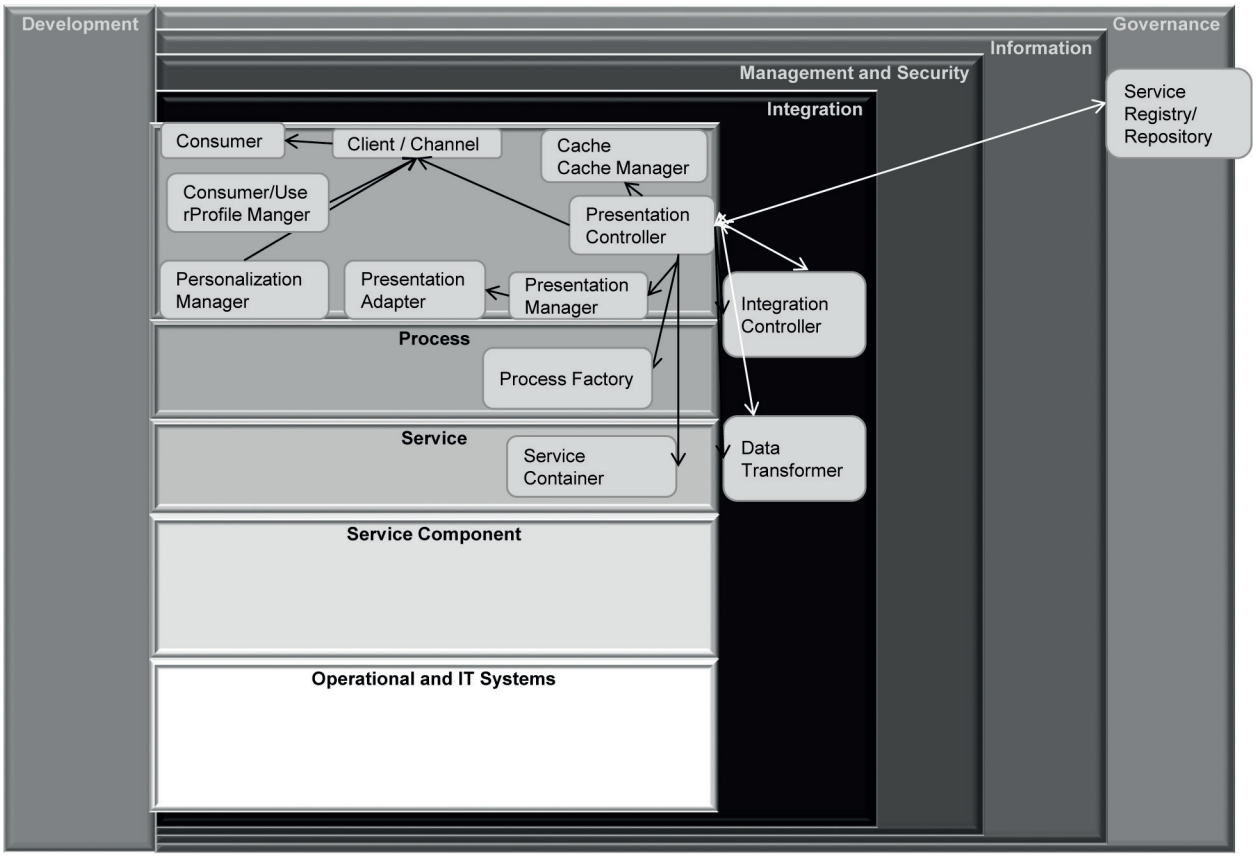


Figure 38 — Key Interactions of the Consumer Layer with Horizontal Layers

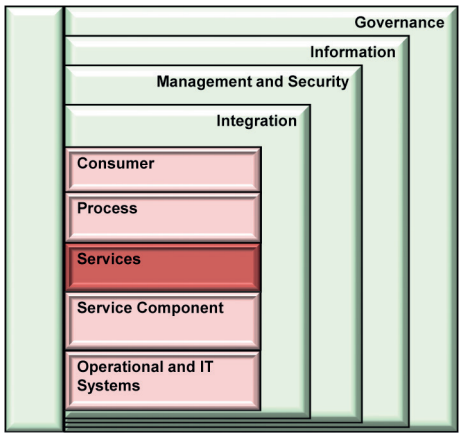
12.5 Usage Implications and Guidance

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

13 Integration Aspect

13.1 Overview

13.1.1 Summary



(From 7.5.7) That Integration Aspect enables the loose coupling between the request and the concrete provider by matching the Service Request and Service Implementation. This loose coupling provided by the Integration Aspect is not only a technical loose coupling addressing protocols, binding, locations or platforms but can also be a business semantic loose coupling performing required adaptations between service requester and provider.

There are numerous sets of capabilities the Integration Layer supports to overcome structural and semantic mismatches at service interfaces. For example, the Integration Aspect supports the integration with solution platforms by the other layers in the SOA RA using mediators, transformers, or adapters to enable the access of services by other ABBs, layers and the capabilities associated with service transport. Mediation includes transformation, routing, and protocol conversion. Integration includes adapters and service enablement. Routing includes service interaction and service virtualization. Transport includes service messaging and message processing. It can be thought of as the plumbing that interconnects the SOA solution and can be driven by policy.

Integration support includes enabling and providing the capability to mediate between the service requester and the service provider. It provides capabilities to transform, route, and convert protocols, enabling support for heterogeneous environment, adapters, service interaction, service enablement, service virtualization, service messaging, message processing and transformation.

Routing support includes capabilities through which a consumer/requestor can connect to the correct service provider. These can start with point-to-point capabilities for tightly coupled endpoint integration and cover the spectrum to a set of intelligent routing, mediation, and other transformation mechanisms often associated with, but not limited to, mediation services provided by an Enterprise Service Bus (ESB). The service description specifies a location where a service is provided and an associated binding and for part of a service contract. A mediation service, on the other hand, provides a location independent mechanism for integration, service substitution, and virtualization.

Transportation support that occurs here is primarily the integration of service component, service, and Process Layers (the “functional” layers). 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, CRM client (used by Customer Service Rep), etc. The transformation of response to HTML (for web), Voice XML (for IVR), or XML string can be done via XSLT functionality supported through mediation service transformation capability in the Integration Aspect.

13.1.2 Context and Typical Flow

The Integration Aspect is a key enabler for an SOA as it provides the capability to mediate, integrate, route and which includes transformation, routing, and protocol conversion to transport service requests

from service requester to the correct service provider. Mediation includes transformation, routing, and protocol conversion. Integration includes adapters and service enablement. Routing includes service interaction and service virtualization. Transport includes service messaging and message processing.

As shown in the UML sequence diagram in Figure 39, the Integration Aspect

— provides a level of indirection between the consumer of functionality and its provider. A service consumer interacts with the service provider via the Integration Aspect. Hence, each service interface is only exposed via the integration aspect (e.g. ESB); never directly and point-to-point integration is done at the Integration Aspect instead of consumers/requestors doing it themselves, and

— consumers and providers are decoupled; this decoupling allows integration of disparate systems into new solutions.

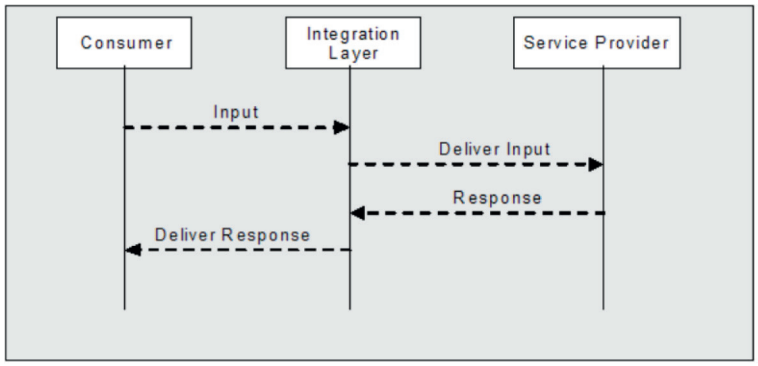


Figure 39 — Usage of the Integration Aspect

13.1.3 Capabilities

There are multiple set of categories of capabilities that the Integration Aspect needs to support in the SOA RA. These categories are as follows.

— **Communication, Service Interaction, and Integration**: This category of capabilities provides the ability to route requests to correct the provider after necessary message transformation and protocol conversion and to connect the service requestor to the service provider and its underlying solutions platforms realizing the requested service. It also provides the ability to discover services and, at runtime, to support the virtualization of services so that changes to the endpoints (or locations from where the services are called and where the services are provided) can occur without impact to service consumers and service providers.

— **Message Processing**: This category of capabilities provides the ability to perform the necessary message transformation to connect the service requestor to the service provider and to publish and subscribe messages and events asynchronously. Message processing capabilities often need to leverage other capabilities like discovery, communication, service interaction and integration.

— **Quality of Service**: This category of capabilities supports handling of transactions and exceptions and other NFRs.

— **Security**: This category of capabilities helps in enforcement of access privileges and other security policies.

— **Management**: This category of capabilities provides the ability to maintain service invocation history and monitor and track the service invocations.

This layer features the following capabilities:

— **Communication, Service Interaction, and Integration**

1) ability to take a service call and messages to the endpoint, i.e. to enable a service consumer to connect/interact with service providers

2) ability to handle service request and service response

3) ability to support communication through a variety of protocols

4) ability to support variety of messaging styles such as one-way, pub-sub, request-response

5) ability to route messages to the correct service provider

6) ability to transform protocol formats, e.g. from SOAP/HTTP to SOAP/Message Queue or SOAP/JMS

7) ability to link a variety of systems that do not directly support service-style interactions so that a variety of services can be offered in a heterogeneous environment

8) ability to store and forward messages using message queuing

— **Message Processing**

9) ability to transform data formats, e.g. from proprietary to standard format or industry standards and *vice versa*

10) ability to transform semantic mapping (data positional mapping)

11) ability to aggregate (including messages and data) from different services and service providers

12) ability to propagate the events from producers to consumers

— **Quality of Service**

13) ability to handle transactions from the other layers, especially when a statically composed service invokes a service chain

14) ability to handle exceptions raised in the process of service invocation and message passing

— **Security**

15) ability to authenticate/authorize for service invocation and message routing

— **Management**

16) ability to capture and record message routing and service invocation history

17) ability to track and monitor the message routing and service invocation activities

18) ability to configure the Integration Aspect

13.1.4 Structural Overview of the Layer

The Integration Aspect ABBs are logically partitioned into categories which support the

— ability to provide the integration of services with underlying solutions platforms, to discover services and their endpoints (or locations from where the services are called and where the services are provided) can occur without impact to service consumers and service providers,

— ability to support mediation; mediation can be thought of as the routing of the request to correct providers after necessary message transformation and protocol conversion and aggregation of the content from different service providers, from different formats and protocols into a common canonical form (data format), while not normative, it is customary to support service messages in an XML format after mediation,

— ability to support different service standards such as WS-Security (see Reference [19]), etc.,

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

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

— ability to perform message transformation,

— ability to support Quality of Service (QoS) requirements such as transaction management, performance criteria, exception handling, etc.,

— ability to support security requirements, and

— ability to track, monitor, and manage service invocations.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 13.1.3 that the ABB supports. For example, in Figure 40, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:'). For example, in the Figure 40, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. MaS: Access Controller supports capability number 15 '15. ability to authenticate/authorize for service invocation and message routing'.

Figure 40 illustrates the ABBs partitioned into key categories.

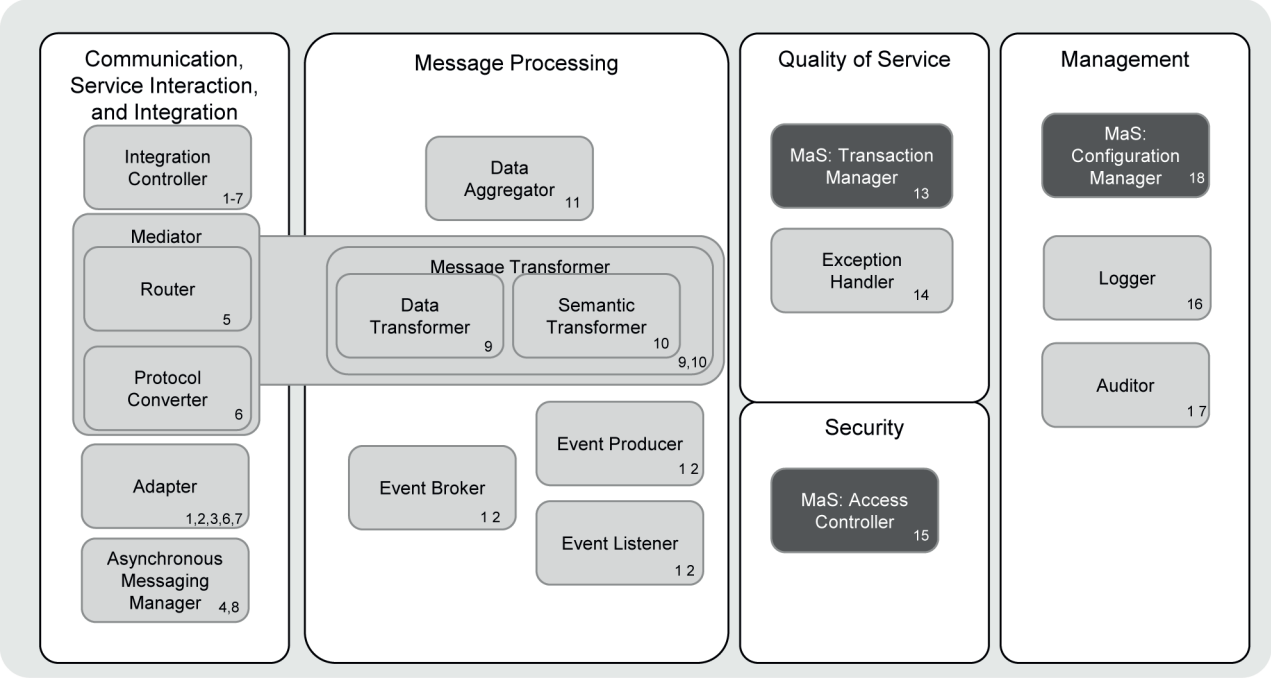


Figure 40 — ABBs in the Integration Aspect

13.2 describes each of the ABBs in detail in the Integration Aspect in terms of their responsibilities. The ABBs are grouped by the capabilities.

13.2 Details of ABBs and Supported Capabilities

13.2.1 Communication, Service Interaction and Integration

13.2.1.1 Integration Controller/Integration Gateway

This ABB represents 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 the Router ABB through the Mediator ABB to service requests and it delegates to the Message Transformer ABB for the data, format and message transformation needs of the other layers.

13.2.1.2 Mediator

This ABB represents the coordinating and handling the service request/response interaction. There is a set of ABBs used to overcome structural and semantic mismatches at service interfaces; therefore, the mediator ABB also supports the transformation between message formats, conversion of protocols, and routing of service call/messages to the service provider. It uses the Data Transformer ABB and, optionally, the Semantic Transformer ABB for the transformations. Finally, it supports static service composition to orchestrate and chain services or system calls or messages together in order to compose services statically. It uses ABBs with more specialized mediation capabilities such as Router, Message Transformer, and the Data Aggregator to do this (which can also be used directly).

13.2.1.3 Router

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

13.2.1.4 Protocol Converter

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

13.2.1.5 Adapter

This ABB represents the interfacing/connectivity of SOA RA layers of a solution to external systems and components and taking a call (message) to the endpoint. In particular, it addresses any necessary mediation (router, message transformer, protocol converter) of the elements outside the Integration Aspect 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.

13.2.1.6 Asynchronous Messaging Manager

This ABB represents the ability to store and forward messages potentially using a message queue and it provides the underlying plumbing to transport messages between service invokers and providers, as well as the ability to store and forward the messages. It supports both reliable transport and guaranteed

delivery. Message queuing is a common example of a mechanism to support asynchronous messaging. A common feature of this ABB is to support message delivery guarantees and reliability.

13.2.2 Message Processing

13.2.2.1 Data Aggregator

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

13.2.2.2 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. This ABB may also attach data or metadata to indicate the details of the message transformation or to enable the messaging infrastructure Semantic Transformer.

13.2.2.3 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 Aspect 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.

13.2.2.4 Data Transformer

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

13.2.2.5 Event Broker

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

13.2.2.6 Event Producer

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

13.2.2.7 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.

13.2.2.8 Quality of Service

13.2.2.9 Management and Security Aspect: Transaction Manager

See 14.2.4.10.

13.2.2.10 Exception Handler

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

13.2.3 Security

13.2.3.1 Management and Security Aspect: Access Controller

See 14.2.2.8.

13.2.3.2 Management

13.2.3.3 Logger

This ABB is responsible for capturing and recording message routing and service invocation activities. It logs data to monitor system exceptions and system stability (data such as resource availability, etc.). This should be interoperable and integrate with the Management and Security Aspect monitoring features. It is important from a monitoring and support for compliance perspective. Policies and granularity of logging should consider that there is a tradeoff between the amount of information logged and performance of the SOA solution.

13.2.3.4 Auditor

This ABB is responsible for tracking and monitoring the message routing and service invocation activities. It supports the capture of audit data, the conversion to standard formats such as XDAS and CBE, the encryption of that data during transport, and the obfuscation of sensitive data. Policies and granularity of auditing should consider that there is a tradeoff between the amount of information recorded and performance of the SOA solution.

13.2.3.5 Management and Security Aspect: Configuration Manager

See 14.2.8.1.

13.3 Inter-Relationships between the ABBs

Figures 41 and 42 show the inter-relationships and a non-normative sequence of interaction. The final, prescriptive sequence of interaction will be defined by the underlying solution architecture and the standards that the invoking other SOA RA layers support.

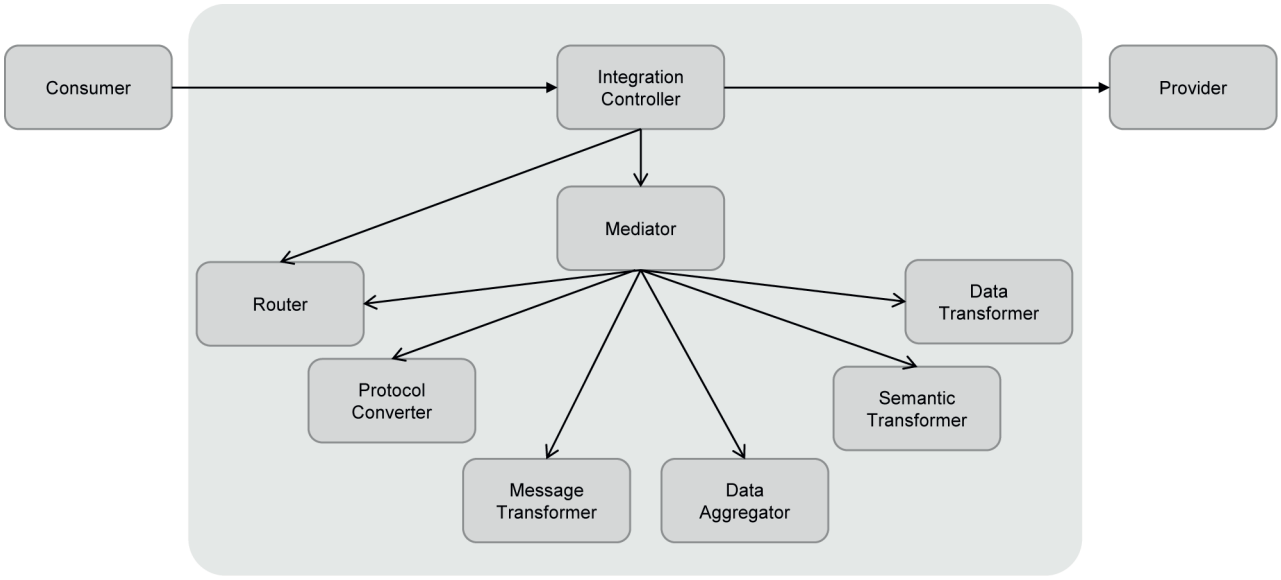


Figure 41 — Simple Interactions between Consumer and Provider through the Integration Aspect

One purpose of the Integration Aspect is to handle integration between disparate providers and consumers. This can be done by the Integration controller, so the consumer invokes a service with invokes the integration controller ABB. This ABB, in turn, uses a Mediator ABB to coordinate handling the request from the consumer. In this case, the mediator the Protocol Converter and Message Transformer ABBs to handle any changes in the message on the wire, then the Data aggregator and Adapter ABBs perform data and semantic changes. Now, the Mediator uses the Router ABB to send the message to the Provider.

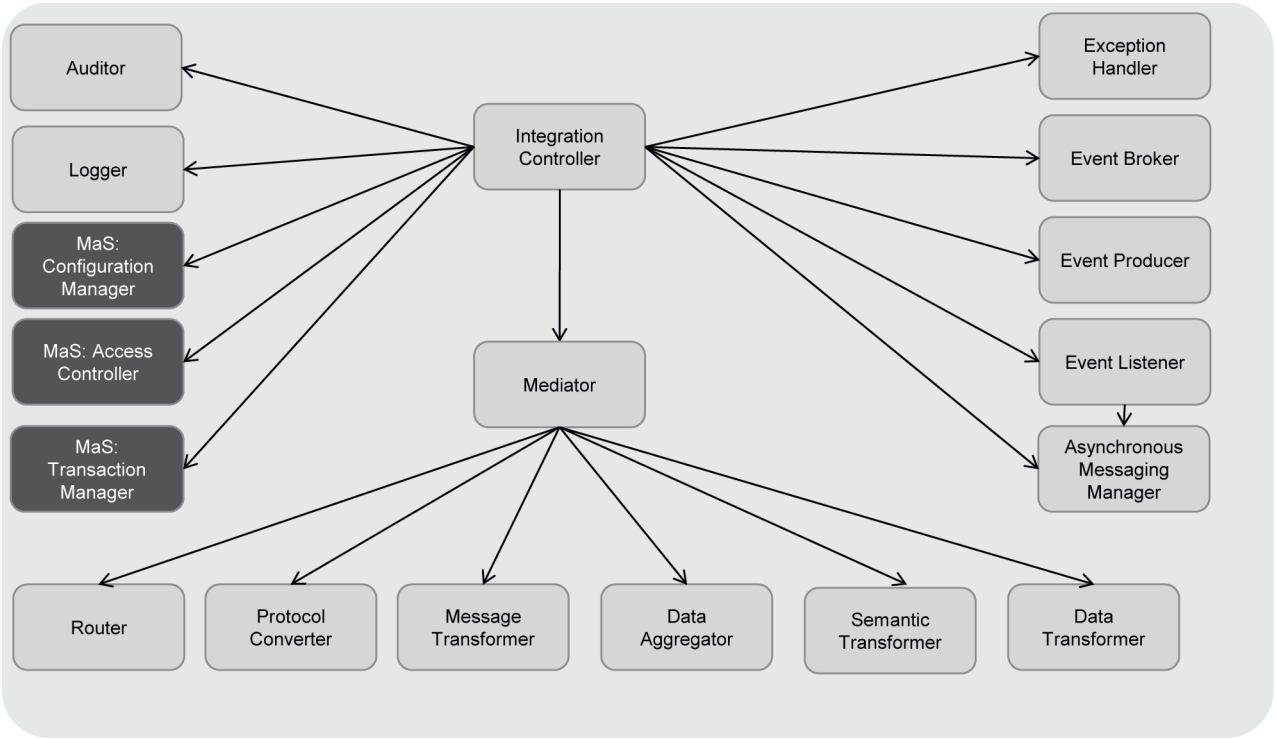


Figure 42 — Relationships among ABBs in the Integration Aspect

In Figure 42, it shows that Integration Aspect does integration and mediation for all kinds of messaging and events in the solution, including auditing, logging, management, security, transactions. Integration

Controller can use any of the ABBs used by the Mediator ABB directly without going through the Mediator. In this case, the Integration Controller is not invoked by a consumer but it is invoked by an Auditor, Logger, Manager, Access Controller or Transaction Manager which needs to be integrated with disparate targets of these messages. The Integration Controller and Mediator can adapt the message to a target that is be synchronous (Event Listener) or asynchronous (Asynchronous Messaging Manager), generates other Events with the Event Producer or Event Broker, or even creates Exceptions with the Exception Handler.

13.4 Significant Intersection Points with other Layers

13.4.1 Interaction with Cross-Cutting Aspects

The Integration Aspect relies on other cross-cutting aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

It relies on the Development Aspect for the following capabilities:

— ability to implement and test integration capabilities including loggers, auditors, event managers, event listeners with tools;

— ability to create and use service descriptions, contracts, and deployment descriptions which can be used to advertise and access the integration capabilities that might be part of the functional interface of the service of the service, like event subscription;

— ability to performance test and tune integration capabilities.

It relies on the Governance Aspect for the following capabilities:

— ability to store metadata for policies;

— ability to support management (storage, retrieval, etc.) of rules to support rules associated with decision points in service mediation, orchestration, and composition; the Integration Aspect will leverage a common business rules capability that can be used by the ESB (the component in the integration aspect which mediates – routes and transforms data);

— ability to determine service endpoints for service virtualization.

It relies on the Management and Security Aspect for the following capabilities:

— ability to authenticate/authorize for service invocation and messages.

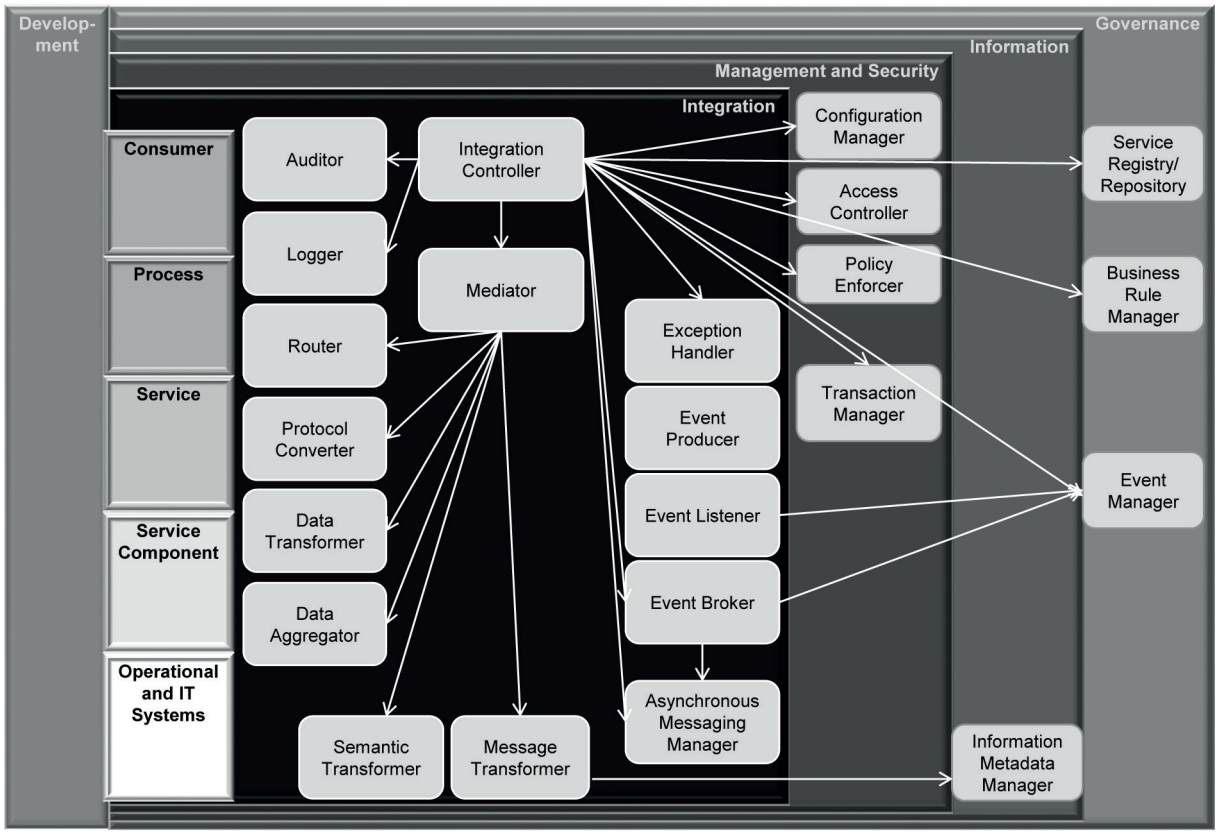


Figure 43 — Key Interactions of the Integration Aspect with Cross-Cutting Aspects

Therefore, the Integration Aspect interfaces with the following ABBs of cross-cutting aspects of the architecture to provide its capabilities.

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

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

— It leverages the Access Controller ABB in the Management and Security Aspect for authenticating/authorizing facility for service invocation and message routing. It also leverages the Policy Enforcer ABB in the Management and Security Aspect to enforce policies local to the Integration Aspect.

— The Data Transformer ABB in the Integration Aspect uses metadata from the Information Aspect and leverages the Information Metadata Manager ABB from the Information Aspect for data transformation.

— The Event Broker ABB, Event Listener ABB, and Event Producer ABB in the Integration Aspect use the Event Manager ABB in the Governance Aspect for event definition and related information.

13.4.2 Interaction with Horizontal Layers

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

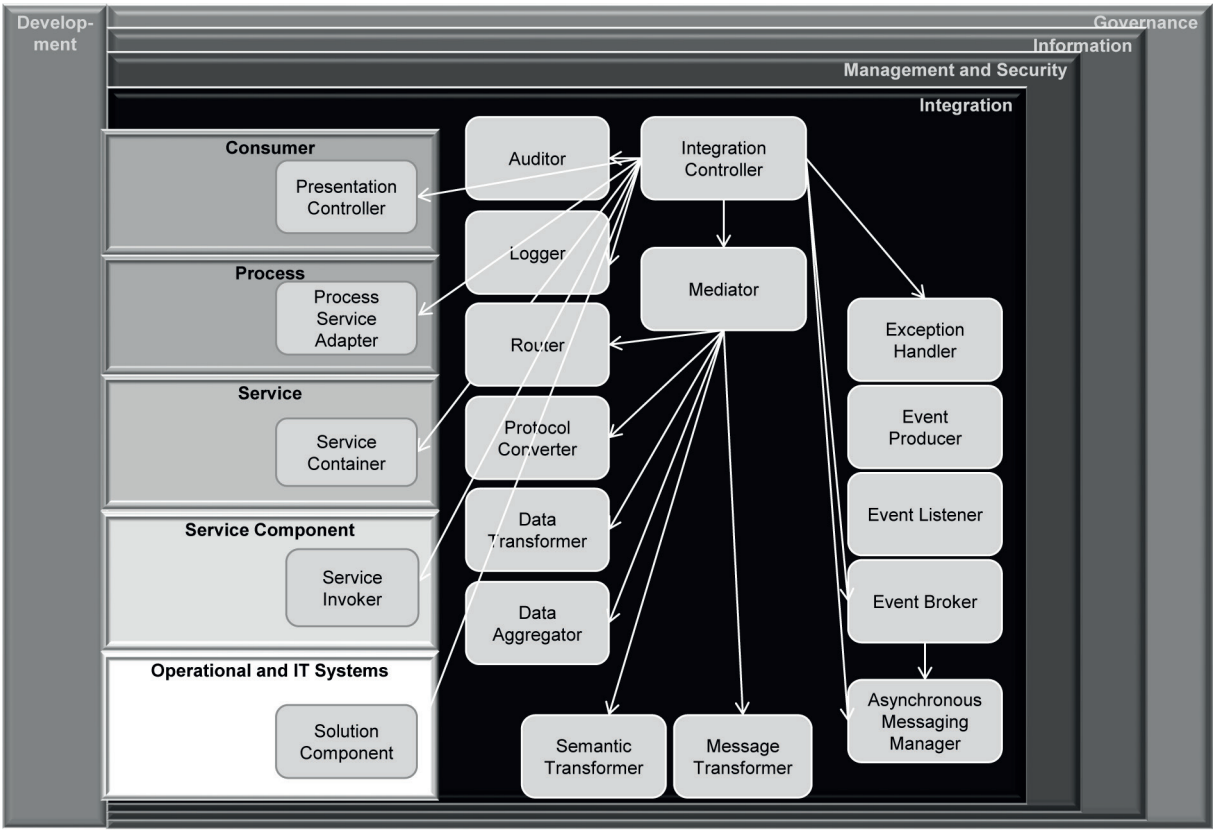


Figure 44 — Key Interactions of the Integration Aspect with Horizontal Layers

Figure 44 shows how a solution can be architected using the ABBs in the horizontal layers of the SOA RA and how they interact with the Integration Aspect.

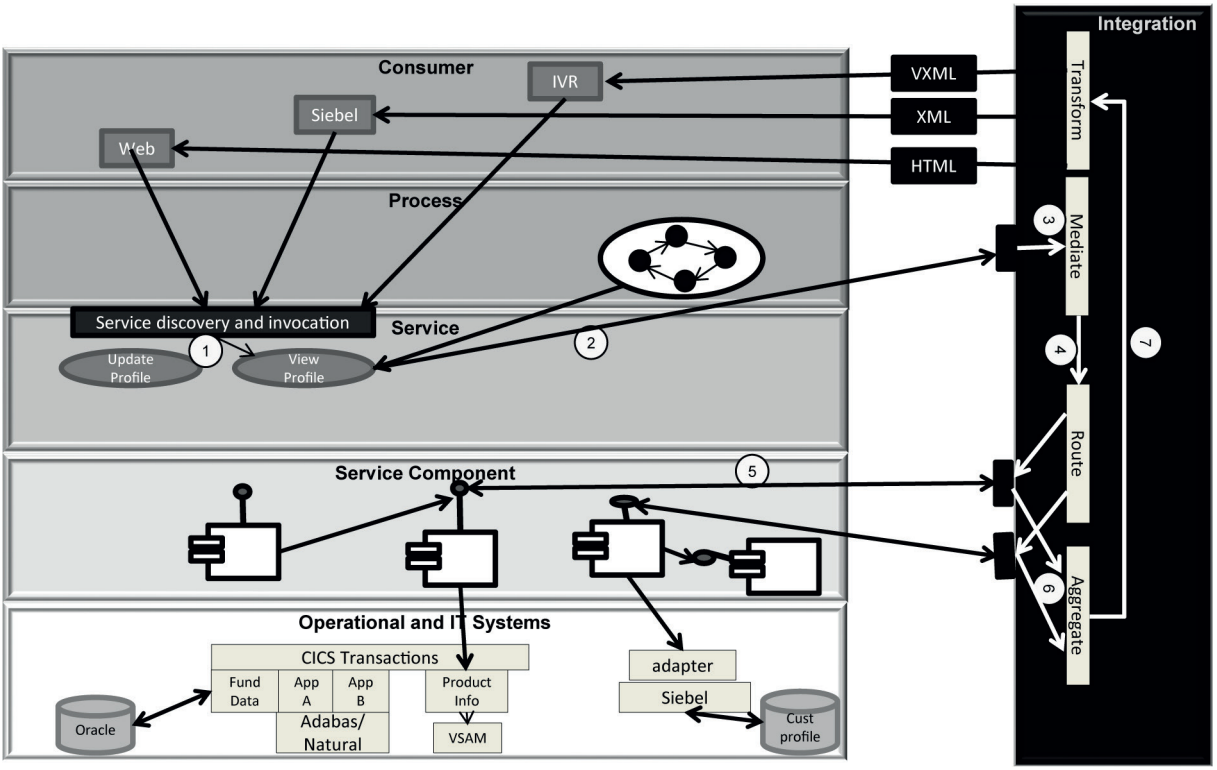


Figure 45 — Detail Interactions of the Horizontal Layers with the Integration Aspect

In the example above, the circled numbers show the steps in the following flow.

a) Through the service discovery and invocation process, the web client looks up the View Profile Service.

b) A connection is made to the Integration Aspect.

c) The ESB performs protocol conversion, if necessary.

d) It subsequently routes the call to the appropriate destination.

e) It then receives the results of the call.

f) It then aggregates the results of the call.

g) The aggregated result is transformed and returned to the client in a format that can be consumed by it (for example, in the case of the web client, the aggregated result is returned in an HTML format)

13.5 Usage Implications and Guidance

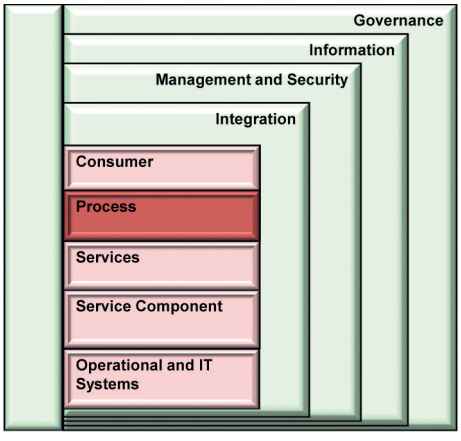
The Integration Aspect also incorporates the support for service virtualization using static routing rules or dynamically at runtime using a Service Registry/Repository. Runtime binding decouples the location of services for consumers of services. Services are exposed to consumers through a Registry/Repository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

To summarize, the Integration Aspect 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 Management and Security (MaS) Aspect

14.1 Overview

14.1.1 Summary



(From 7.5.8) The Management and Security Aspect supports non-functional requirement (NFR) related issues as a primary concern of SOA and provides a focal point for dealing with them in any given solution. It contains the capabilities for ensuring that a SOA meets its requirements with respect to: monitoring, reliability, availability, manageability, transactionality, maintainability, scalability, security, safety, life cycle, auditing and logging, etc. It includes the same scope as the traditional Fault, Configuration, Accounting, Performance, and Security (FCAPS) from ITIL (see Reference [27]) or Reliability, Availability, Serviceability (RAS) (see Reference [28]).

Management and Security is especially important for SOA solutions to enable the loosely coupled solutions to scale and efficiently fulfil non-functional requirements. This layer provides capabilities that maintain and ensure the "quality of service (QOS)". To enable both Management and Security, this layer provides capabilities that

— provides solution management of various concerns, such as availability, reliability, security, and safety, as well as the mechanisms to support, track and monitor, and manage solution quality controls,

— provides the ability to monitor and enforce a multitude of policies and corresponding business rules including business level policies, security policies, access privileges and data access policies,

— serves as an observer of the other layers and can create events when a non-compliance condition is detected or (preferably) when a non-compliance condition is anticipated,

— provides the service and SOA solution lifecycle processes with the capabilities required to ensure that the defined policies, non-functional requirements (NFRs), and governance regimens are adhered to for service and SOA solution lifecycle processes,

— supports the ability to monitor and manage at both the business level [in terms of key performance indicators (KPIs), events and activities in business processes] and the IT systems level (for the security, health and well-being of IT systems, services, applications, networks, storage and processors), and

— supports monitoring and capturing of service and solution metrics in an operational sense and signalling of non-compliance with non-functional requirements relating to 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.

Finally, the same kinds of management and monitoring that apply to businesses are important for managing services and SOA solutions and may need extensions to handle the service-oriented nature and the cross domain boundaries of many SOA solutions. These traditional capabilities supported by SOA solutions include

— IT Systems Monitoring and Management,

— Service and SOA Solution Monitoring and Management,

— Business Activity Monitoring and Management,

— Event Management,

— Configuration and Change Management,

— Policy Monitoring and Enforcement,

— Lifecycle management, and

— Auditing and Logging.

SOA security addresses the protection of the SOA solution against threats across the vulnerability dimensions of a service-oriented architecture. This includes protecting the interactions between service consumers and service providers, as well as protecting all of the elements that contribute to the architecture. Examples of threats to be protected from are destruction, corruption, removal, disclosure, and interruption. Some of the security dimensions to help protect against these threats include access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy.

The capabilities that explicitly address security are as follows.

— Security Management: Manages and monitors security and secure solutions. This provide ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, enable the IT organization to manage IT-related risks and compliance, and provide the automation and audit basis for security management.

— Facilities Security Management: This category of capabilities provides the command centre for security management, as well as the operational security capabilities for non-IT assets and services to ensure protection, response, continuity and recovery. It also supports for security of physical assets such as locations, facilities, services, inventory, physical access control, human identity, etc.

Important areas for policy enforcement are security, auditing, messaging transportation, infrastructure availability, service availability and reliability. Responses (dispensations and appeals) to non-compliance and exceptions are also defined by the Governance Aspect.

14.1.2 Context and Typical Flow

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

— monitoring and management both at the business level in terms of Key Performance Indicators (KPIs), events, and business activities in the business processes and at the IT systems level for the security, health, and wellbeing of IT systems, services, applications, networks, storage, and compute servers, and

— monitoring and enforcement of a multitude of policies and corresponding business rules including business-level policies, security policies, access privileges, data access policies, etc.

This layer provides capabilities to enable solution quality including availability, reliability, security, and safety. It also includes mechanisms to support, track, monitor, and manage solution quality control.

The Management and Security Aspect provides capabilities needed to support the service and SOA solution lifecycle processes so that they can ensure that the defined policies, Non-Functional Requirements (NFRs), and governance regimens are adhered to.

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

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

In the SOA RA policies, business rules and the NFRs and policies for the SOA solution are defined and captured in the Governance Aspect but are monitored and enforced in the Management and Security Aspect. Important areas of policy enforcement are security, messaging transportation, and infrastructure availability and service availability. This layer also supports security management and systems management for SOA solutions. Responses (dispensations and appeals) to non-compliance and exceptions are defined by the Governance Aspect as well.

14.1.3 Capabilities

There are multiple sets of categories of capabilities that the Management and Security Aspect needs to support in the SOA RA. These categories are as follows.

— **Facilities Security Management**: This category of capabilities provides the command centre for security management, as well as the operational security capabilities for non-IT assets and services to ensure protection, response, continuity, and recovery. It also supports security of physical assets such as locations, facilities, services, inventory, physical access control, human identity, etc. This is especially important for SOA because providers and consumers may be contracting with external parties in ecosystems to provide services and management and security characteristics and certification might be necessary. Requirements by consumers for transparency of facilities management has been seen with Cloud Computing.

— **Security Management**: This category of capabilities manages and monitors security and secure solutions. This provides the ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, address how software, systems, and services are developed and maintained throughout the software lifecycle, maintain the security status through proactive changes reacting to identified vulnerabilities and new threats, enable the IT organization to manage IT-related risks and compliance and provide the automation basis for security management.

— **IT Systems Monitoring and Management**: This category of capabilities provides monitoring and management of IT infrastructure and systems. This includes the ability to monitor and capture metrics and status of IT systems and infrastructure.

— **Services and SOA Solution Monitoring and Management**: This category of capabilities provides monitoring and management of software services and applications. This includes the ability to capture metrics and to monitor and manage application and solution status.

— **Business Activity Monitoring and Management**: This category of capabilities provides monitoring and management of business activities and business processes. It provides the ability to analyse this event information, both in real-time/near real-time, as well as stored (warehoused) events, and to review and assess business activities in the form of event information and determine responses or issues alerts/notifications.

— **Event Management**: This category of capabilities provides the ability to manage events and enables the complex event processing in the SOA RA.

— **Policy Monitoring and Enforcement**: This category of capabilities provides a mechanism to monitor and enforce a multitude of policies and corresponding business rules including business-level policies, security policies, access privileges, and data access policies. This provides the ability to find and access policies, evaluate and enforce policies at checkpoints or on metrics captured, signal and record compliance status or metrics, send notification and log of non-compliance and change rules, policies, configuration, and status.

— **Configuration and Change Management**: This category of capabilities provides the ability to change solution configuration and descriptions.

— **Registry and Repository**: This category of capabilities provides the ability to store and access policies, rules and other data needed to enable security and management.

This layer features the following capabilities:

— **Facilities Security Management**

1) Ability to ensure protection, response, continuity, and recovery

2) Ability to approve authority for security

3) Ability to ensure that physical and operational security is maintained for locations, assets, humans, environment, and utilities

4) Ability to provide surveillance and monitoring of locations, perimeters, and areas

5) Ability to enforce entry control

6) Ability to provide for positioning, tracking, and identification of humans and assets; continuity, and recovery operations

7) Ability to secure physical assets, such as locations, facilities, services, inventory, physical access control, human identity, etc.

8) Ability to ensure the safety of a solution from types of failure, damage, error, accidents, and harm as defined by the governance aspect

— **Security Management**

9) Ability to ensure appropriate authentication based on defined roles and/or prescribed attributes

10) Ability to ensure appropriate authorization based on defined roles and/or prescribed attributes

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 consider and address, as appropriate, 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 parts of software development and deployment

18) Ability to maintain the security status through proactive changes to the system, reacting to identified vulnerabilities and new threats, and through responding to detected incidents and reported problems

19) Ability to identify, quantify, assess, and report on IT-related risks that contribute to the enterprise's operational risk by providing all services to analyse and report security information and security events, and create alarms and insight

20) Ability to provide the automation basis for security management

21) Ability to provide and enforce policies for access control

22) Ability to control access to individual data items in messages

— **IT Systems Monitoring and Management**

23) Ability to monitor, manage, and configure IT systems hardware, including operating systems which are part of an SOA solution

24) Ability to monitor, manage, and configure IT network hardware systems which are part of an SOA solution

25) Ability to monitor, manage, and configure IT storage hardware systems which are part of an SOA solution

— **Service and SOA Solution Monitoring and Management**

26) Ability to coordinate overall Quality of Service requirements for the SOA solution

27) Ability to describe Quality of Service NFRs

28) Ability to manage solutions and services from solution delivery to solution termination

29) Ability to handle transactions from the other layers, especially when a statically composed service invokes a service chain

30) Ability to capture metrics such as the percentage of executions that the solution does not fail and the percentage of executions of the solution that execute within a prescribed period of time

31) Ability to capture metrics such as the metric for percentage of time that the solution is invokable

32) Ability to capture metrics such as the metric for the response time of network access to a service or solution

33) Ability to react to infrastructure changes to maximize availability

34) Ability to log or report on availability metrics

35) Ability to evaluate the availability metrics against the NFRs (policy)

36) Ability to capture metrics on performance of services and solutions

37) Ability to change configuration and policy to ensure meeting SLAs

38) Ability to change configuration and policy to ensure performance optimization

39) Ability to support virtualization of resources to support performance optimization

40) Ability to record, track, and monitor the cost of executing a specific solution

41) Ability to monitor current status of the solution

42) Ability to change the current status of the solution

43) Ability to check quality of service requirements for valid status

44) Ability to issue events for non-compliance to quality of service requirements

45) Ability to measure, gather, evaluate, and test metrics against policies on a regular basis

— **Business Activity Monitoring and Management**

46) Ability to analyse this event information, both in real-time/near real-time, as well as stored (warehoused) events

47) Ability to review and assess business and service activity in the form of event information and determine responses or issue alerts/notifications

— **Event Management**

48) Ability to interface with the Integration Aspect and obtain events from the Integration Aspect

49) Ability to control issuance of events in the solution

50) Ability to send or issue events indicating non-compliance to Quality of Service requirements

51) Ability to subscribe to events issued by the solution

52) Ability to log events and business messages

53) Ability to control logging frequency and size

— **Policy Monitoring and Enforcement**

54) Ability to check QoS requirements for valid rules

55) Ability to change rules to comply with QoS requirements

56) Ability to change QoS requirements to comply with rules

57) Ability to send events for non-compliance to QoS requirements

58) Ability to evaluate policies and their effects

59) Ability to recognize, respond to, and resolve conflicts between policies

60) Ability to assess and enforce compliance with policies

61) Ability to automatically respond to violations of policies (enforce)

62) Ability to enable policy enforcement

63) Ability to discover, analyse, transform, distribute, evaluate, and enforce security policies

64) Ability to manage non-functional quality of service solution requirements from solution delivery to solution termination

65) Ability to manage lifecycle of policies

66) Ability to represent policies

67) Ability to author policies

68) Ability to manage instances of policies

69) Ability to change policies

70) Ability to disable, discard, and discontinue policies

71) Ability to monitor and capture metrics and status

72) Ability to find and access policies

73) Ability to evaluate policies at checkpoints or on metrics captured

74) Ability to automate monitoring for violations of policy

— **Configuration and Change Management**

75) Ability to capture configuration (authoring tools)

76) Ability to change configuration

77) Ability to check QoS requirements for valid configurations

78) Ability to change configuration to comply with QoS requirements

79) Ability to send events for non-compliance to QoS requirements

80) Ability to track and record changes, configuration, metadata, policy, etc. happening in the solution

81) Ability to recover from or even reverse changes made to the solution

82) Ability to ensure that changes are executed in compliance with relevant governance policies

83) Ability to change metadata, including service descriptions

84) Ability to propagate metadata changes to other repositories and descriptions

— **Registry and Repository**

85) Ability to store QoS policies and rules

86) Ability to locate/find/return QoS policies and rules

14.1.4 Structural Overview of the Layer

The ABBs in the Management and Security Aspect can be thought of as being logically partitioned into categories which support

— ability to provide the command centre for security management, as well as the operational security capabilities for non-IT assets and services,

— ability to manage and monitor security and secure solutions,

— ability to monitor and manage IT infrastructure and systems,

— ability to monitor and manage software services and applications,

— ability to monitor and manage business activities and business processes and associated KPIs,

— ability to manage events,

— ability to monitor and enforce a multitude of policies and corresponding business rules,

— ability to change solution configuration and descriptions, and

— ability to store and access policies and business rules.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 14.1.3 that the ABB supports. For example, in Figure 46, the ABBs from the Governance Aspect are a dark grey (with a prefix of 'Governance:'), while the ABB from the Integration Aspect is shown as black (with a prefix of "Integration"). For example, in Figure 46, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'Governance:'. Governance: Change Control Manager supports capability number 80, 81, and 82: '82. Ability to ensure changes are executed in compliance with relevant governance policies'. The Integration: Event Producer supports capability 50-57 '50 is: 50: Ability to send or issue events indicating non-compliance to quality of service requirements'.

Figure 46 illustrates the ABBs partitioned into key categories.

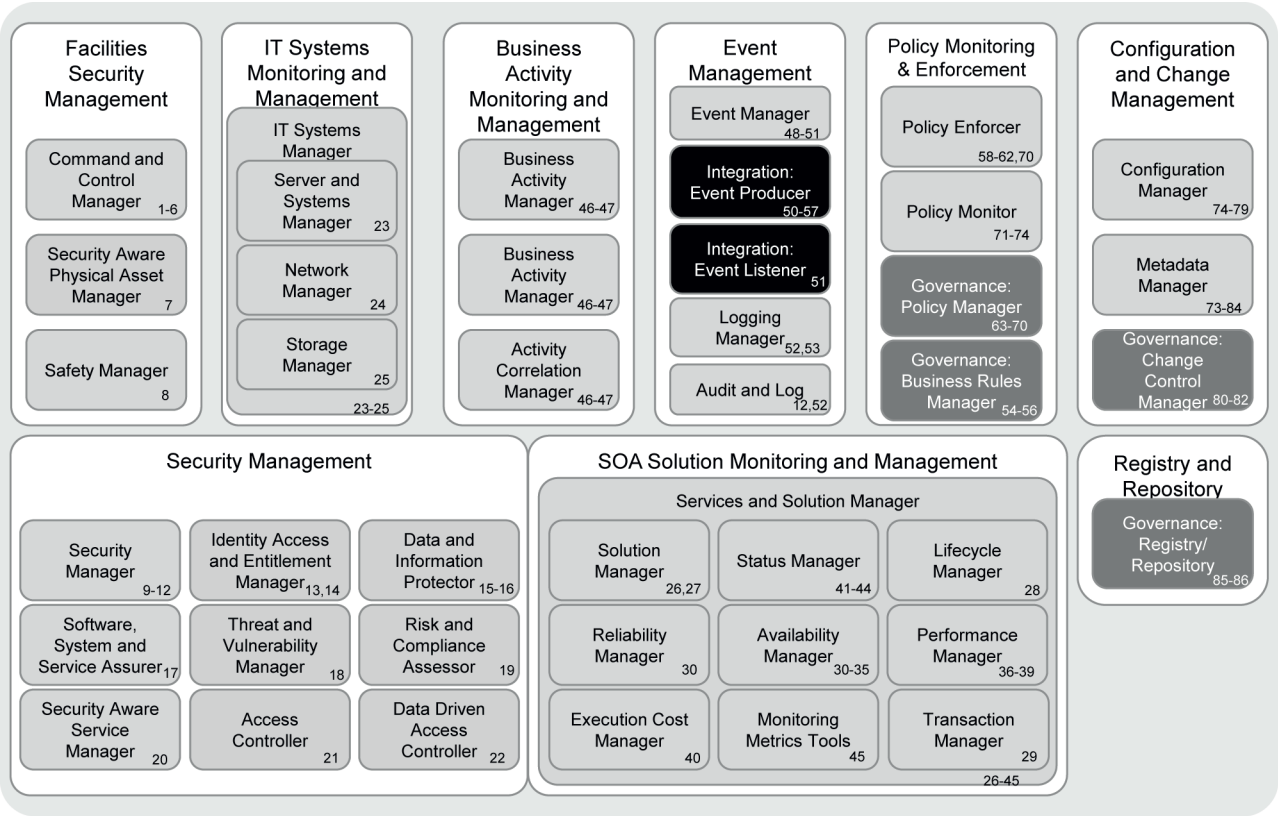


Figure 46 — ABBs in the Management and Security Aspect

14.2 describes in detail each ABB in terms of its responsibilities and organizes them by the capability category.

14.2 Details of ABBs and Supported Capabilities

14.2.1 Facilities Security Management

14.2.1.1 Command and Control Manager

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

— providing the approving authority for security,

— ensuring that physical and operational security is maintained for locations, assets, humans, environment, and utilities,

— providing surveillance and monitoring of locations, perimeters, and areas,

— enforcing entry controls, and

— providing for positioning, tracking, and identification of humans and assets, continuity and recovery operations.

14.2.1.2 Security-Aware Physical Asset Manager

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

14.2.1.3 Safety Manager

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

14.2.2 Security Management

14.2.2.1 Security Manager

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

14.2.2.2 Identity, Access, and Entitlement Manager

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

14.2.2.3 Data and Information Protector

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

14.2.2.4 Software, System, and Service Assurer

This ABB represents an actor to consider and address as appropriate 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 parts of software development and deployment. Its responsibilities include

— structured design process,

— threat modeling,

— risk assessment,

— design reviews for security,

— source code review,

— source code analysis,

— dynamic application analysis,

— source code control,

— access monitoring,

— code/package signing and verification,

— quality assurance testing,

— supplier and third-party code validation, and

— security problem and incident management of software and services.

14.2.2.5 Threat and Vulnerability Manager

This ABB represents the capability 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. Its responsibilities include vulnerability testing, vulnerability scanning, virtual patching, threat analysis, and risk analysis.

14.2.2.6 Risk and Compliance Assessor

This ABB represents the capabilities for IT organizations to identify, quantify, assess, and report on IT-related risks that contribute to the enterprise's operational risk by providing all services to analyse and report security information and security events, and create alarms and insight. Its responsibilities include security and compliance dashboard, forensics, reporting, specially, risk aggregation and reporting, compliance audit.

14.2.2.7 Security-Aware Service Manager

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

14.2.2.8 Access Controller

This ABB represents a kind of Policy Enforcer ABB providing access control and enforcing policies related to access control and rights. This includes the enforcement of "trust" policies such as authentication/authorization facilities for service invocation and message routing, as well as access privileges for various participants to data. It typically supports authorization and authentication functionalities for registered participants, including federated authentication (single sign-on) and the ability to ensure that the appropriate audit logging is carried out. This ABB depends on the Governance Aspect, which provides the ABBs that support defining security policies, to retrieving security policies and acting as a local policy decision point and local Policy Enforcement Point (PEP). It can include support for standards such as SAML (authentication and authorization), XDAS, and CBE (audit and logging). It leverages the Identity, Access, and Entitlement Manager ABB to fulfil its responsibilities.

14.2.2.9 Data-Driven Access Controller

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

14.2.3 IT Systems Monitoring and Management

14.2.3.1 IT Systems Manager

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

14.2.3.2 Server and Systems Manager

This ABB represents the systems manager of the runtime environment.

14.2.3.3 Network Manager

This ABB represents the capabilities for monitoring network infrastructure performance, proactively identifying potential network issues and problems, and isolating and correcting network faults.

14.2.3.4 Storage Manager

This ABB represents the capabilities for managing storage resources, including access to local, networked, and virtualized storage.

14.2.4 SOA Solution Monitoring and Management

14.2.4.1 Services and Solution Manager

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

14.2.4.2 Solution Manager

This ABB represents the centre of the Management and Security Aspect. It coordinates the management of the solution and all of the other ABBs. This ABB is responsible for coordinating solution lifecycle, security, availability, configuration, and change.

14.2.4.3 Status Manager

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

14.2.4.4 Lifecycle Manager

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

14.2.4.5 Reliability Manager

This ABB represents the capabilities for measuring and reporting the reliability of a solution. Reliability managers may take actions to ensure that reliability objectives are met.

14.2.4.6 Availability Manager

This ABB represents the capabilities for measuring and reporting the availability quality of a solution. Availability managers may take actions to ensure that availability objectives are met. 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.

14.2.4.7 Performance Manager

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

14.2.4.8 Execution Cost Manager

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

14.2.4.9 Monitoring Metric Tools

This ABB represents the capabilities to measure, gather, evaluate, and test 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.

14.2.4.10 Transaction Manager

This ABB represents the capabilities to manage 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 (see Reference [20])], with WS-Atomic Transaction (see Reference [21]) for atomic, ACID transactions and WS-BusinessActivity (see Reference [22]) 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.

14.2.5 Business Activity Monitoring and Management

14.2.5.1 Business Activity Manager

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

14.2.5.2 Business Activity Monitor

This ABB represents the capabilities to monitor the event, business activities in a business processes, and services. It interfaces with the Integration Aspect to handle notification and propagation of events.

14.2.5.3 Activity Correlation Manager

This ABB represents the capabilities for reviewing and assessing inbound business and service activity in the form of event information and determines responses or issues alerts/notifications.

14.2.6 Event Management

14.2.6.1 Event Manager

This ABB represents the capabilities for controlling 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.

14.2.6.2 Integration Aspect: Event Producer

See 13.2.2.6.

14.2.6.3 Integration Aspect: Event Listener

See 13.2.2.7.

14.2.6.4 Logging Manager

This ABB represents the capabilities for configuring and enabling logging of events and business messages. Logging frequency and log size should be configurable. Policies and granularity of logging should consider that there is a tradeoff between the amount of information recorded and performance of the SOA solution.

14.2.6.5 Audit and Log

This ABB represents the capabilities for persisting events and business messages in a log or in an audit log.

14.2.7 Policy Monitoring and Enforcement

14.2.7.1 Policy Enforcer

This ABB represents the Policy Enforcement Points (PEPs) in the architecture for enforcing QoS policies and security policies across all the functional layers and cross-cutting aspects.

The Policy Enforcer interacts with the Governance Aspect 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.

This ABB may execute diverse policies associated with lower-level message functionality such as addressing transformation, routing, caching, compression, and other content handling functions provided by integration aspect and other layers in the SOA RA. The place where policies are actually applied and enforced changes depending on the lifecycle stage, for example, registry/repository, message transports and IT management systems. This ABB representing the PEPs can be software or hardware-based high-performance functional component that intercepts, inspects, filters, and performs content-aware policy-driven processing on application messages and their payloads.

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

14.2.7.2 Policy Monitor

This ABB represents the capabilities for enabling automation of monitoring for violations of policy. It includes checkpoints in SOA processes and is an integral part of compliance processes policy. This ABB obtains its policies from the Policy Manager ABB in the Governance Aspect. This ABB is passive

and interacts with the Policy Enforcer ABB to take any actions when violations are detected. It is responsible for

— capturing real-time collection and statistical analysis for display,

— providing a management console for visibility into the management of distributed network of PEPs and the status of these enforcements,

— logging and aggregating measurements and highlighting significant events, and

— correlating, analysing, and visualization of data fed in by the policy enforcer ABB at various PEPs.

14.2.7.3 Governance Aspect: Policy Manager

See 16.2.4.4.

14.2.7.4 Governance Aspect: Business Rules Manager

See 16.2.3.2.

14.2.8 Configuration and Change Management

14.2.8.1 Configuration Manager

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

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

— through template-based configuration, where a user can select a specific template based on the corresponding service request scenario. The system will select all the rules associated with this template and configure the ABBs to support the rules. This requires scenario templates to be created and stored in a registry/repository to be selected when needed;

— through dynamic template creation, where a user selects certain characteristics and the system will determine the appropriate rules and configure using relevant ABBs at runtime. For instance, a user may require that the system adopt an industry messaging standard and satisfy some SLAs. Based on these requirements, the system during runtime will select the appropriate data transformation, protocol conversion, and service providers that meet the SLAs.

14.2.8.2 Metadata Manager

This ABB is responsible for managing metadata in repositories.

14.2.8.3 Governance Aspect: Change Control Manager

See 16.2.6.4.

14.2.9 Registry and Repository

14.2.9.1 Governance Aspect: Repository/Repository

See 16.2.2.1.

14.3 Inter-Relationships between the ABBs

Figure 47 illustrates the key relationships among the ABBs in the Management and Security Aspect for management of the solution in an operational environment.

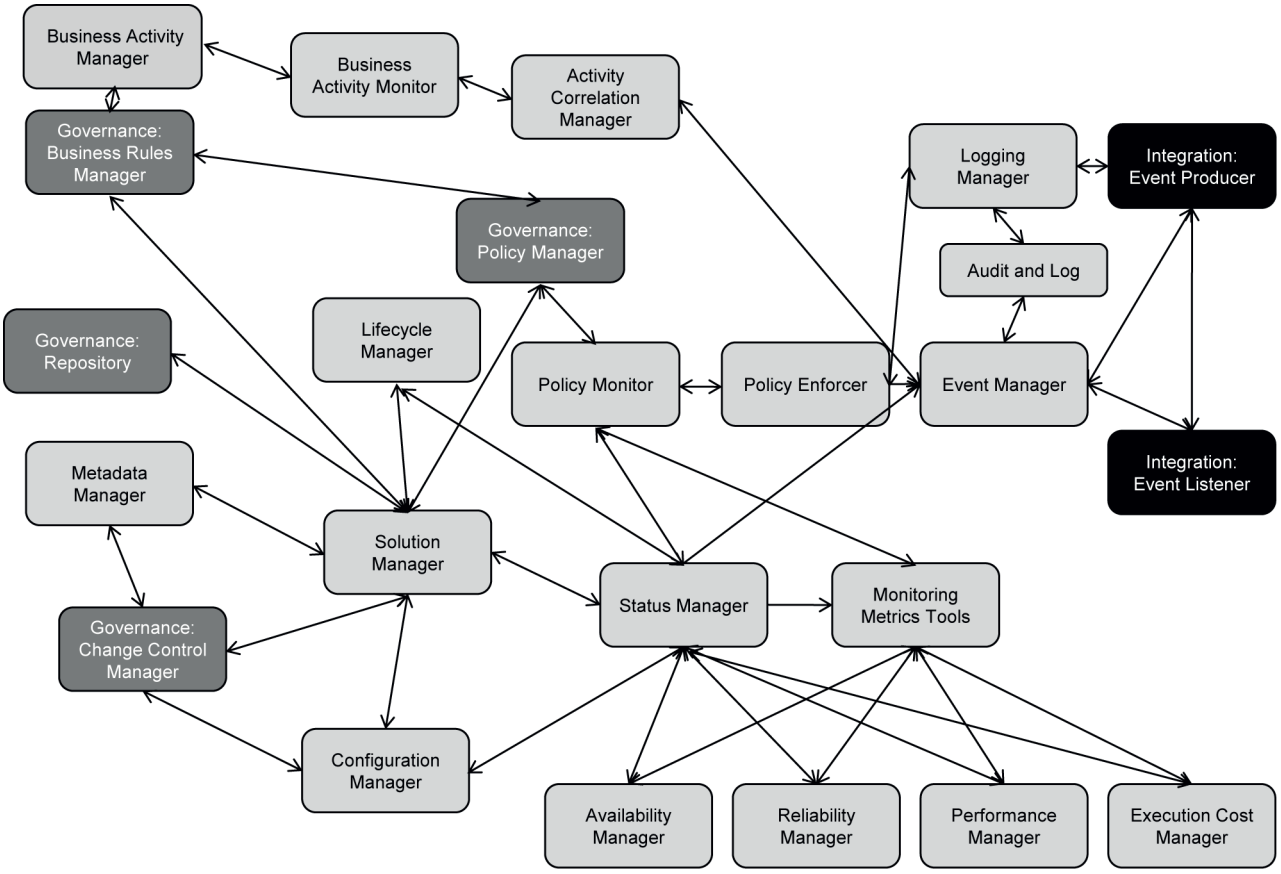


Figure 47 — Relationships among ABBs in the Management and Security Aspect

Figure 47, the solution manager coordinates the management of SOA solution. It gets its high level policies and resource information from the Business Rules Manager, which is where the business rules for the business and SOA solution are updated and retrieved from. The Business Rules Manager updates the Policy Manager for the solution, as well as the Business Activity Manager which drives the Business Activity Monitor. The Business Activity Correlation Manager takes information from the monitor and creates business events using the Event Manager.

The Solution Manager retrieves policies it needs to manage the solution from Policy Manager and resource and service information from the Registry/Repository. Based on these policies, the configuration is updated using the Configuration Manager and the Change Control Manager. When changes are made to the configuration or the solution resources, the Metadata Manager is used to update the metadata for the Solution. The Solution Manager also feeds those metadata changes to the Registry/Repository when it is appropriate.

To make resources or service available to consumers, the Solution Manager coordinates the lifecycle of the solution through the Lifecycle Manager using the Status Manager. The Status Manager brings resources 'on-line' using the Availability Manager. The Availability Manager also monitors the availability of the solution at runtime and reports to the Monitoring Metrics Tools.

At runtime, the Monitoring Metrics Tools, based on policies from the Policy Manager, also uses the Reliability Manager to monitor and report on the reliability of the solution, performance of the solution, and execution cost for the SOA solution. The Monitoring Metrics tools, as defined by policy, sends information and alerts back to the Policy Manager sends it to the Policy Enforcer. The Policy Enforcer can update local policy based on the new information if needed; these changes in policy may drive the

Solution Manager to make changes in the solution using the Configuration Manager, Monitoring Metrics Manager and Lifecycle Manager. The Policy Enforcer also logs significant changes using the Logging Manager in the solution and send events using the Event Manager to interested parties who are using appropriate Event Listener. Sometimes, based on Policy, the Logging Manager can send messages as events via the Event Producer to the Event Listener and Event Manager.

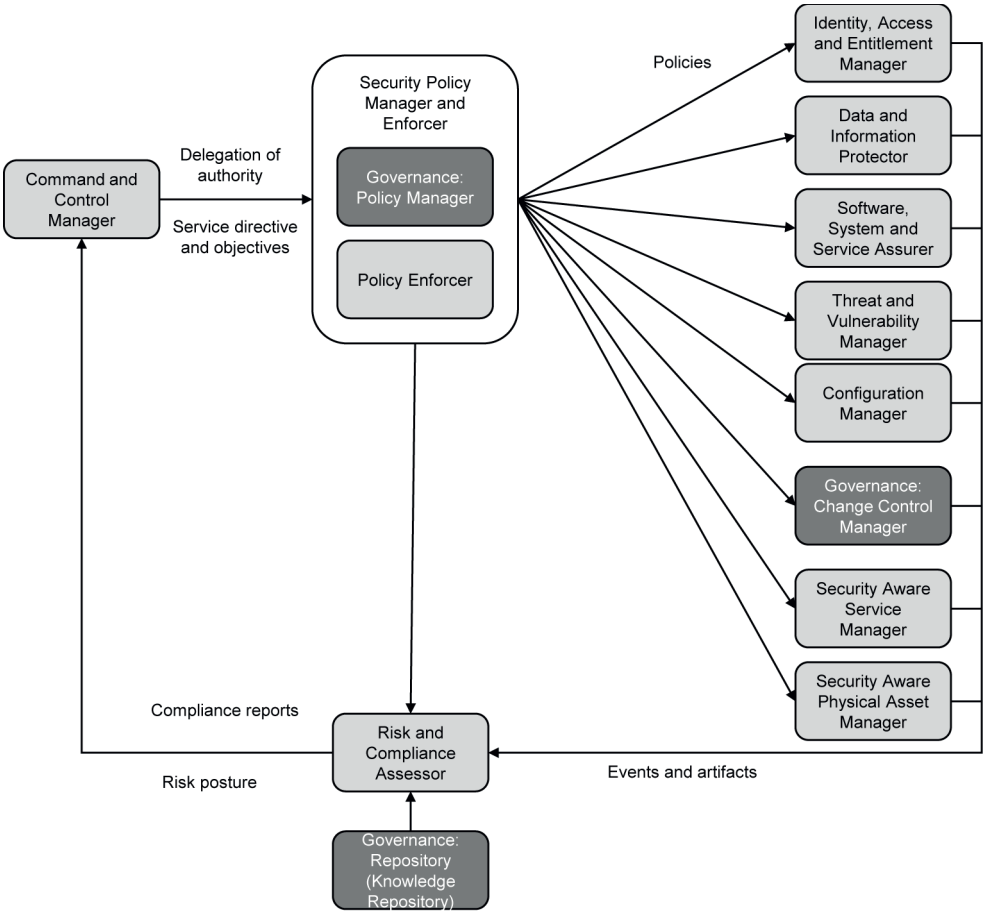


Figure 48 — Relationships among ABBs for Command and Control Management and Security Management in the Management and Security Aspect

Figure 48 illustrates interactions between the ABBs for security and Control. In this example, the Risk and Compliance Assessors assess the physical and cyber risks to the organization using information from the Registry/Repository which is acting as a Knowledge Repository. The Assessor determines the risk posture and documents it as policies and business rules based on business objectives. The Assessor also defines what compliance reports are necessary from the solution. All of this information is sent to the Command and Control Manager which sends the information and then delegates enforcement to the Security Policy Manager and Enforcer. The Security Policy Manager and Enforcer updates the security policy using the Policy Manager and uses the Policy Enforcer to ensure the policies are being respected. These Policies drive and are enforced by the Identity Access and Entitlement Manager, Data Information Protector, Software System and Service Assurer, Threat and Vulnerability Manager, Configuration Manager, Change Control Manager, Security Aware Service Manager, and Security Aware Physical Asset Manager. All of these managers collaborate and send events and reports to the Risk and Compliance Assessor.

14.4 Significant Intersection Points with other Layers

14.4.1 Interaction with Cross-Cutting Aspects

The Management and Security Aspect depends on other cross-cutting aspects in the SOA RA to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

— It depends on the Development Aspect for implementation and testing of MaS capabilities, including Event Managers, Status Managers, and Metrics reporting. It also uses the performance and testing tools, as well as load simulators to enable the Performance Manager and Monitoring. Rules and Descriptions editing and processing tools are used to drive the current management and security activities for the SOA solution.

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

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

— It depends on the Information Aspect for definition of events.

The Management and Security Aspect is used by other cross-cutting aspects to fulfil their respective responsibilities; for example:

— the Policy Enforcer ABB is leveraged by the Integration, Information Architecture, and Governance Aspects to enforce multitude of policies for the respective layers;

— the Access Controller ABB is leveraged by the Integration, Information Architecture, and Governance Aspects to enforce security and access control policies for the respective layers;

— the Data-Driven Access Controller ABB is leveraged by Information Aspect to enforce access control policies based on individual data items.

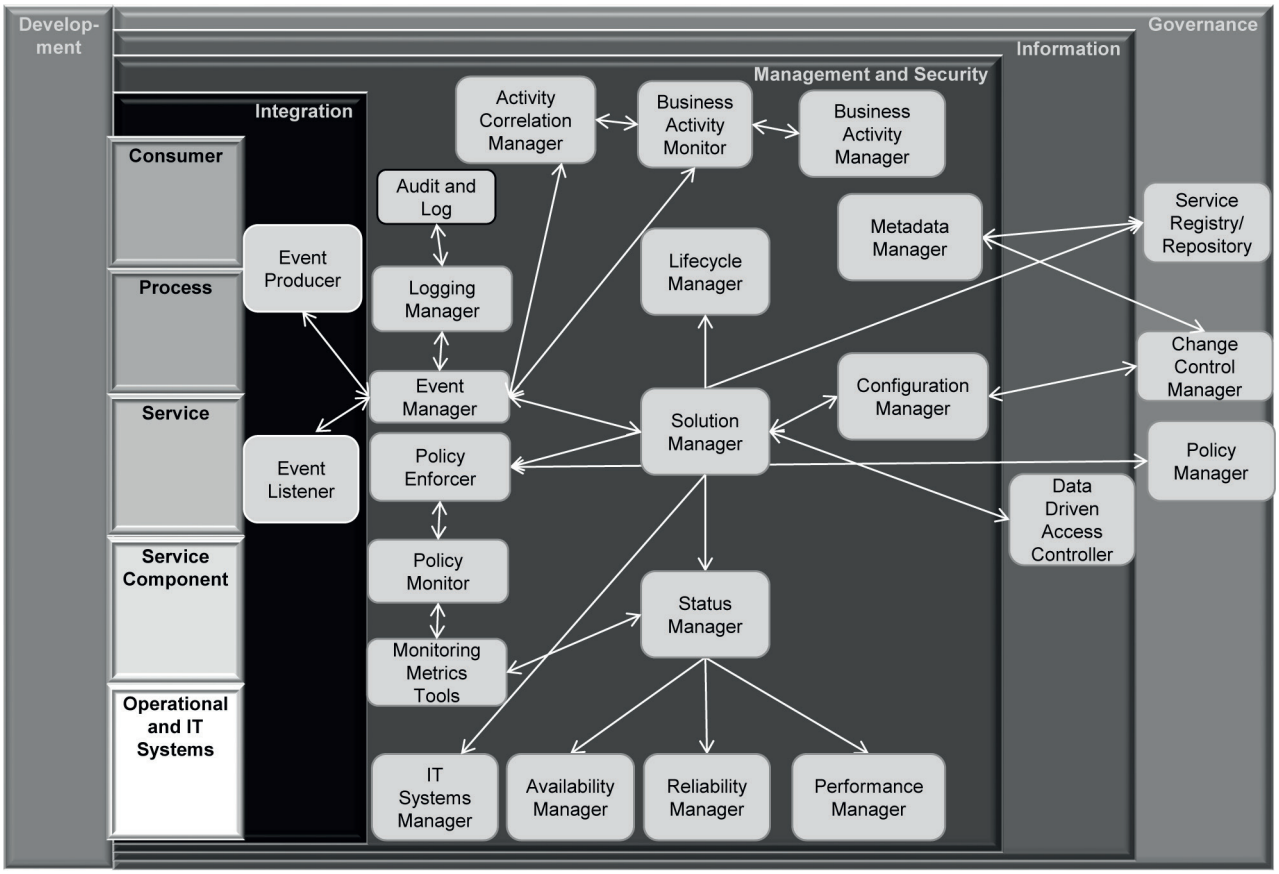


Figure 49 — Key Interactions of the Management and Security Aspect with Cross-Cutting Aspects

14.4.2 Interaction with Horizontal Layers

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

— the Policy Enforcer ABB is leveraged by the Consumer, Business Process, Service, and Service Component Layers to enforce a multitude of policies for the respective layers,

— the Access Controller ABB is leveraged by the Consumer, Business Process, Service, and Service Component Layers to enforce security and access control policies for the respective layers,

— the IT Systems Manager ABB manages all the resources in the Operational and IT Systems Layer,

— the Status Manager ABB is updated by the Service Container when a service changes status, and

— all layers collaborate with the Management and Security Aspect via the Solution Manager ABB which coordinates the QoS and security needs of the SOA solution.

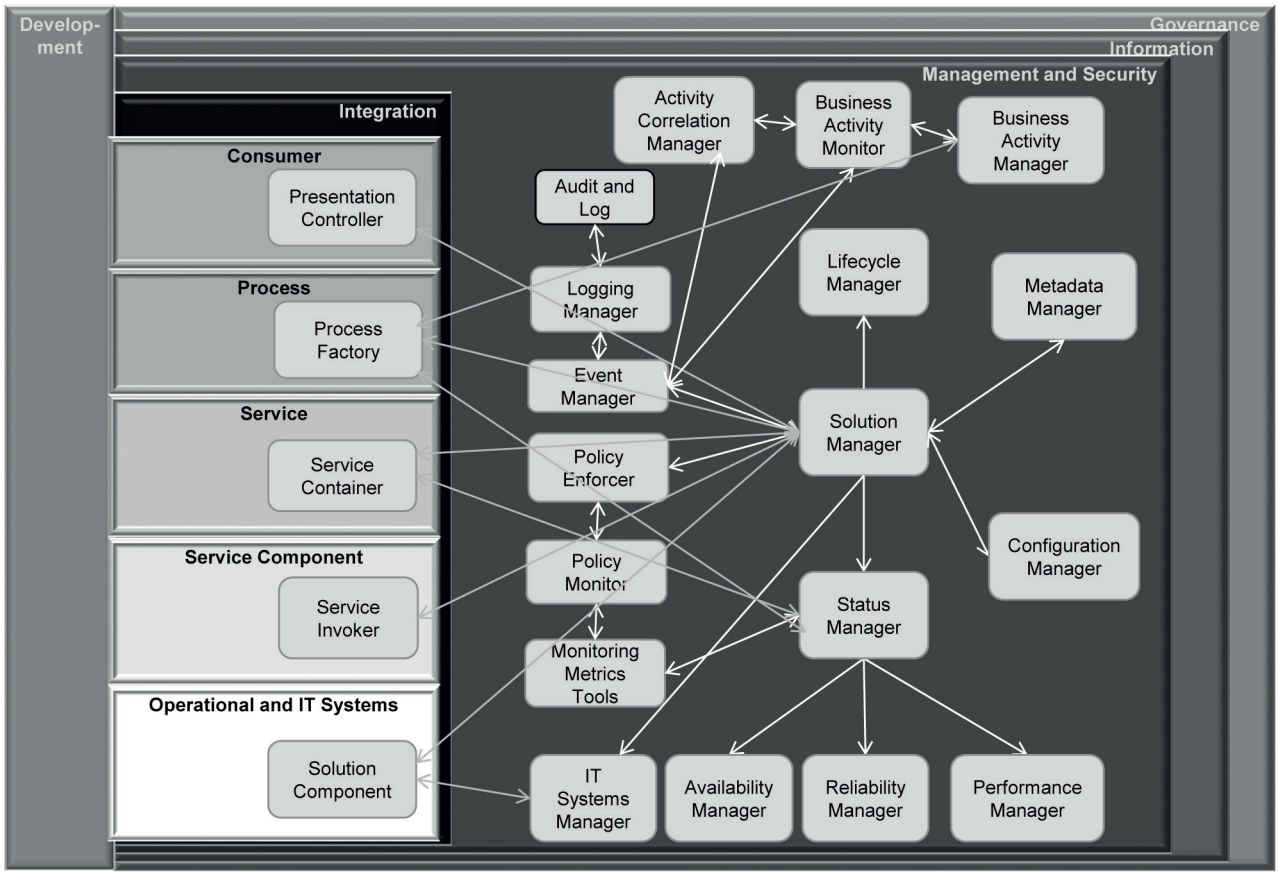


Figure 50 — Key Interactions of the Management and Security Aspect with Horizontal Layers

14.5 Usage Implications and Guidance

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

— reliability,

— availability,

— manageability,

— scalability, and

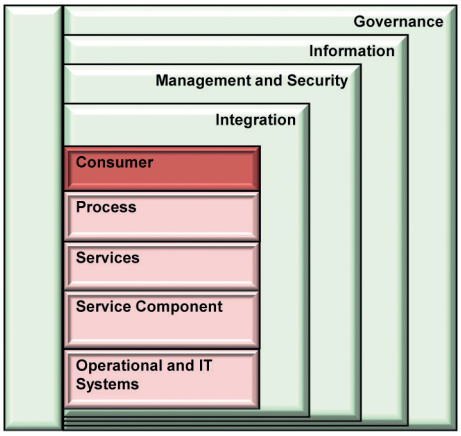
— security

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

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

15 Information Aspect

15.1 Overview



15.1.1 Summary

(From 7.5.9) The Information Aspect provides capabilities for enabling development of a unified representation of the information assets of an organization as represented by its IT services, systems and SOA solutions. The unified representation for an organization may require rationalization and ongoing coordination of assets from across many organizations. Representing information enables business needs and processes to be aligned with the business vocabularies.

The Information Aspect includes information architecture, business analytics and intelligence, and metadata considerations. It focuses on the inclusion of key considerations pertaining to information architectures that can be used as the basis for the creation of business analytics and business intelligence through the analysis of data held in repositories. These repositories include metadata content that is stored using capabilities provided in this layer. The Information Aspect supports the establishment of information services capability architectures that can be used as the basis for the creation of business analytics and business intelligence through data marts and data warehouses. 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 a systematic improvement in data quality.

The Information Aspect supports the following capabilities:

— ability to support information services capability that will support a shared, common and consistent expression of data;

— ability to integrate information across diverse actors and organizations in order to effectively communicate across the different organizational domains;

— ability to define metadata that is used across the SOA RA and, in particular, the metadata that is shared across the layers;

— ability to enable secured and protected information via interaction with the Management and Security Aspect;

— ability to support business activity monitoring and critical to the usage of the SOA RA and its realization.

An information virtualization and information service capability may involve 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 Context and Typical Flow

Associated with the primary objective of this layer are a number of capabilities. This layer includes information architecture, business analytics and intelligence, metadata considerations, and ensures the inclusion of key considerations pertaining to information architectures that can also be used as the basis for the creation of business analytics and business intelligence through data marts and data warehouses.

15.1.3 Capabilities

There are multiple set of categories of capabilities that the Information Aspect needs to support in the SOA RA. These categories are as follows.

— **Information Services**: This category of capabilities addresses the support of information services. Information services provide a uniform way of representing, accessing, maintaining, managing, analysing, 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 (Master Data Manager/MDM) approach with a data manager as a service. The second approach focuses on integrating the appropriate information in a timely and consistent manner, analysing and attempting to improve the quality of data, and ensuring consistency and integrity of business-critical data and facts across the enterprise. This approach is known as the information as a service approach.

— **Information Integration**: This category of capabilities addresses the support of information integration and enables capabilities for information services.

— **Basic Information Management**: This category of capabilities addresses basic information management concerns such as metadata and unstructured data management.

— **Information Security and Protection**: This category of capabilities addresses the support of information security and protection concerns.

— **Business Analytics**: This category of capabilities addresses the support of business analytics and business activity monitoring. It enables organizations to leverage information to better understand and optimize business performance. It supports entry points of reporting to deep analytics and visualization, planning, aligned strategic metrics, role-based visibility, search-based access and dynamic drill-through, and alert and detect in-time actions.

— **Information Definition and Modeling**: This category of capabilities defines fundamental constructs of SOA information and events.

— **Information Repository**: This category of capabilities addresses support of the information registry/repository in order to persist data such as metadata, master data, analytical data, operational data, and unstructured data.

This layer features the following capabilities.

— **Information Services**

1) Ability to expose data as services, to add/remove/manipulate data entries in different services or service components, and to disable some data from outside access

2) Ability to interface with the Integration Aspect in multiple ways such as message-based, service call, batch interface

3) Ability to handle representing data from various data sources in a unified data format, ability to transform and map data from one format to another and align data from different resources

4) Ability to manage the lifecycle of business entities

5) Ability to manage the hierarchy and relationship among data

6) Ability to validate records against defined business rules

7) Ability to validate and enforce data quality rules

8) ability to notify and trigger actions based upon events detected within the data

— **Information Integration**

9) Ability to perform Extract-Transform-Load (ETL capabilities) data from one source to other, ability to extract relevant information from sources, transform the information into the appropriate integrated form, and load the information into the target repository

10) Ability to perform Enterprise Information Integration (EII) capabilities, such as access to federated query to structure and unstructured data

11) Ability to virtualize data representing actual data from the actual data registry/repositories of various types, such as a DB2 database in the Operational and IT Systems Layer, or an Excel

12) Ability to handle data transformation (including transformation of data types and contents) and to aggregate data from multiple data sources

13) ability to perform data standardization and perform data reconciliation including semantic reconciliation

14) Ability to cleanse and match inbound records to existing data

15) Ability to cache data in support of the data virtualization/information services capability

— **Basic Information Management**

16) Ability to manage and maintain metadata in a common metadata registry/repository for the enterprise

17) Ability to capture, aggregate, and manage unstructured content in a variety of formats such as images, text documents, web pages, spreadsheets, presentations, graphics, email, video, and other multimedia

18) Ability to author, configure, manage, customize, and extend metadata

— **Information Security and Protection**

19) Ability to handle access privileges of various participants to data

20) Ability to control access on individual data items

21) Ability to monitor and manage data usages using a log-like facility; typical traceability log includes who has accessed the data, when, and what part of the data has been accessed

— **Business Analytics**

22) Ability to analyse data access history and provide optimization algorithms and business intelligence for data optimization

23) Ability to query and search capabilities for enterprise information

24) Ability to visualize interactively the results from business analytics and data analysis

25) Ability to interface with the Integration Aspect and obtain events from the integration aspect, ability to analyse this event information, both in real-time/near real-time, as well as stored (warehoused) events

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

— **Information Definition and Modeling**

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

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

29) Ability to define business events

— **Information Repository**

30) Ability to store operational and reshaped information (structured and unstructured) that adds business value including common model of data, used to share canonical forms (common data models) between SOA Integration Aspect 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

15.1.4 Structural Overview of the Layer

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

— ability to support information services capability, critical to support a shared, common, and consistent expression of data,

— ability to integrate information across the enterprise or ecosystem in order to enable information services capability,

— ability to define metadata and master data that is used across the SOA RA and, in particular, the metadata that is shared across the layers,

— ability to secure and protect information,

— ability to support business analytics and business activity monitoring critical to the usage of the SOA RA and its realization,

— ability to define information and events that are some of the fundamental constructs of SOA, and

— ability to persist and store data.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 15.1.3 that the ABB supports. For example, in Figure 51, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Integration Aspect is shown as black (with a prefix of "Integration"). For example, in Figure 51, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. MaS: Access Controller supports capability number 19: '19. ability to handle access privileges of various participants to data'. The Integration: Data Transformer supports capability '13: Ability to perform data standardization and perform data reconciliation including semantic reconciliation'.

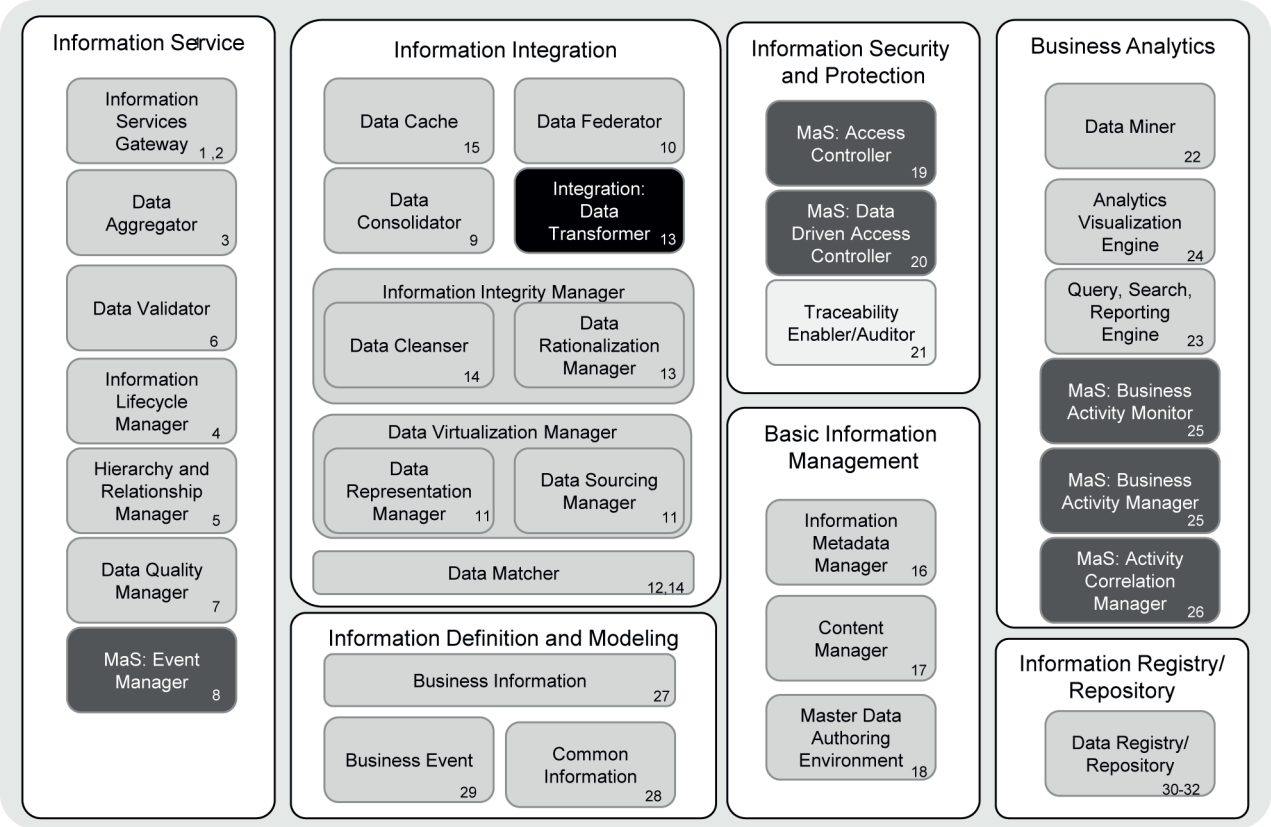


Figure 51 — ABBs in the Information Aspect

15.2 describes each of the ABBs in the Information Aspect in terms of their responsibilities and organizes them by capability category.

15.2 Details of ABBs and Supported Capabilities

15.2.1 Information Service

15.2.1.1 Information Service Gateway

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

— to expose information as a Service;

— to manipulate data entries in different services and service components;

— to control access to certain selected parts of data; disable some data parts from outside access.

This ABB acts as the gateway to the Information Aspect. 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 Aspect and consumers of information services and is critical to expose information as a Service. It provides a consistent entry point to the Information Aspect through multiple mechanisms such as messaging, service calls, and batch processing. This ABB leverages capabilities and ABBs from the Integration Aspect.

15.2.1.2 Data Aggregator

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

Its responsibilities include

— dispatching requests to other ABBs in the Information Aspect,

— invoking the Data Virtualization Manager ABB for handling data transformation (including transformation of data types and contents) and aggregating data from multiple data sources to provide a unified format and model to other ABBs and consumers of information services,

— invoking the Data Validator ABB to validate against business rules,

— invoking the Data Quality Manager ABB for enforcing data quality rules, and

— invoking the Event Manager ABB in the Management and Security Aspect for triggering event notification based on data.

15.2.1.3 Data Validator

This ABB represents capabilities for validating records against defined business rules.

15.2.1.4 Information Lifecycle Manager

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

15.2.1.6 Data Quality Manager

This ABB represents capabilities for validating and enforcing data quality rules, standardizing the data for both value and structure, and performing data reconciliation including semantic reconciliation. It leverages the information integrity ABB to fulfil its responsibilities.

15.2.1.7 Management and Security Aspect: Event Manager

See 14.2.6.1.

15.2.2 Information Integration

15.2.2.1 Data Cache

This ABB represents capabilities for storing data in support of the data for expedited access or when the source of the data is unavailable. 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 concerns.

15.2.2.2 Data Consolidator

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

15.2.2.3 Data Federator

This ABB represents capabilities for providing Enterprise Information Integration (EII) capabilities for federated query access to structured and unstructured data so that large heterogeneous sets of data sources appear as a single homogeneous data source.

15.2.2.4 Integration Aspect: Data Transformer

See 13.2.2.4.

15.2.2.5 Information Integrity Manager

This ABB represents capabilities 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.

15.2.2.6 Data Cleanser

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

15.2.2.7 Data Rationalization Manager

This ABB represents capabilities for performing data rationalizing and reconciliation.

15.2.2.8 Data Virtualization Manager

This ABB represents capabilities for providing virtual access and unified representation of enterprise data sources. It contains, uses and provides functionality for Data Representation Manager and the Data Sourcing Manager for use by the other ABBs.

15.2.2.9 Data Representation Manager

This ABB represents capabilities for handling representation of data from various data sources in a unified data format and for creation of unified views of data. In other words, this ABB intends to hide various data sources and present data in uniform formats to other ABBs for data handling. This ABB may link to various data sources and handle relationships between the data sources. This makes consumers of information services (exposed through the Information Services Gateway) and other ABBs independent of the source and supports consistency in data.

15.2.2.10 Data Sourcing Manager

This ABB represents capabilities for enabling access to different data sources using different protocols. It provides unified access to data in files and represents the actual data registry/repositories in various types, such as a DB2 database in the Operational and IT Systems Layer, or an Excel file databases. It uses an Adapter ABB from the Integration Aspect to provide the ability to integrate with data sources in different solution platforms (external data sources).

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

15.2.2.11 Data Matcher

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

15.2.3 Information Security and Protection

15.2.3.1 Management and Security Aspect: Access Controller

See 14.2.2.8.

15.2.3.2 Management and Security Aspect: Data-Driven Access Controller

See 14.2.2.9.

15.2.3.3 Traceability Enabler/Auditor

This ABB represents capabilities for monitoring and managing data usage using a log-like facility. It interprets log information and stores it in databases to analyse 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. Traceability is enabled and audited for services, solutions and resources. Policies and granularity of tracing should consider that there is a tradeoff between the amount of information traced and performance of the SOA solution.

15.2.4 Business Information Management

15.2.4.1 Information Metadata Manager

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

15.2.4.2 Content Manager

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

15.2.4.3 Master Data Authoring Environment

This ABB represents capabilities 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.

15.2.5 Business Analytics

15.2.5.1 Data Miner

This ABB represents capabilities for enabling data access history analytics, optimization algorithms and business intelligence for data optimization. It enables building of descriptive and predictive models and the use of these models to uncover previously unknown trends and patterns in vast amounts of data from across the enterprise in order to support decision-making.

15.2.5.2 Visualization Engine

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

15.2.5.3 Analytics Query, Search, Reporting Engine

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

15.2.5.4 Management and Security Aspect: Business Activity Monitor

See 14.2.5.2.

15.2.5.5 Management and Security Aspect: Business Activity Manager

See 14.2.5.1.

15.2.5.6 Management and Security Aspect: Activity Correlation Manager

See 14.2.5.3.

15.2.6 Information Definition and Modeling

15.2.6.1 Business Information

This ABB represents the definition of information about the business that is captured and accessed as business vocabularies, business glossaries, terms, and key business entities of the organization and their definition.

15.2.6.2 Common Information

This ABB represents definition of entities and their relationship, logical data definition for database design, and message model for service definition and description.

15.2.6.3 Business Event

This ABB represents definition of business events.

15.2.7 Information Registry/Repository

15.2.7.1 Data Registry/Repository

This ABB represents the registry/repository that stores operational and reshaped information including the following.

— Analytical Data Registry/Repository: Includes operational data stores, data warehouse, data mart, staging areas, and *ad hoc* workspaces.

— Operational Data Registry/Repository: Includes transactional hub, ERP, Supply Chain, CRM, etc.

— Master Data Registry/Repository: Stores instance and definition of master data and historical data that records changes to master data.

— Unstructured Data Registry/Repository: Includes textual objects, graphical objects, multi-media objects, etc.

— Management and Security Aspect: Metadata Registry/Repository: Enables storage of metadata. It stores metadata that describes the master data taxonomies and XML schemas and rules for business logic and data validation. It stores information regarding transformation of data types and content and the ability to aggregate data from multiple sources.

These registry/repositories introduced here represent logical entities that may be implemented as separate repositories, a federation of such registry/repositories, or as a consolidated registry/repository.

15.3 Inter-Relationships between the ABBs

The relationship among these ABBs is shown for different scenarios. All of the following scenarios use an Information Service Gateway (1) to access the right ABBs for the request. The first scenario is for Information as a Service, where information is retrieved from multiple sources.

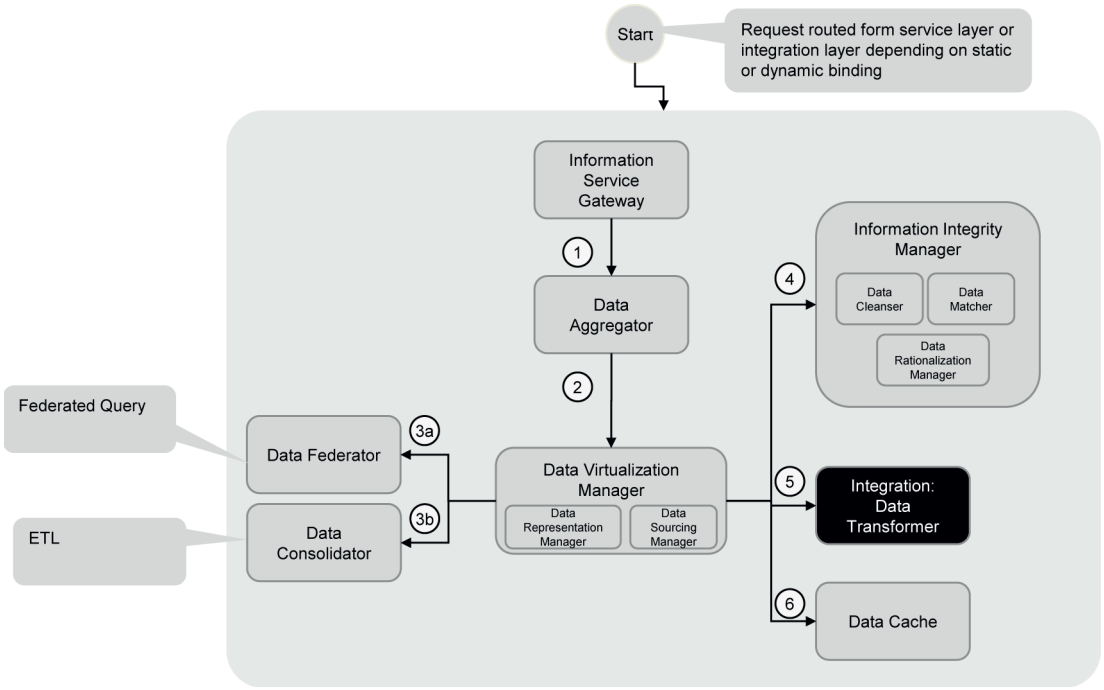


Figure 52 — Key Interactions among ABBs in the Integration Aspect in a Query Scenario

In this scenario in Figure 52, the Data Aggregator (2) uses the Data Virtualization Manager to coordinate the interactions needed to retrieve information. In this case, the information to be offered is first retrieved from multiple sources using Data Consolidators and Data Federators (3a, 3b), then the data is cleansed and rationalized using the ABBs in the Information Integrity Manager (4), then the data is transformed (5) and cached (6) before it is returned to the service requester via the Information Service Gateway.

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

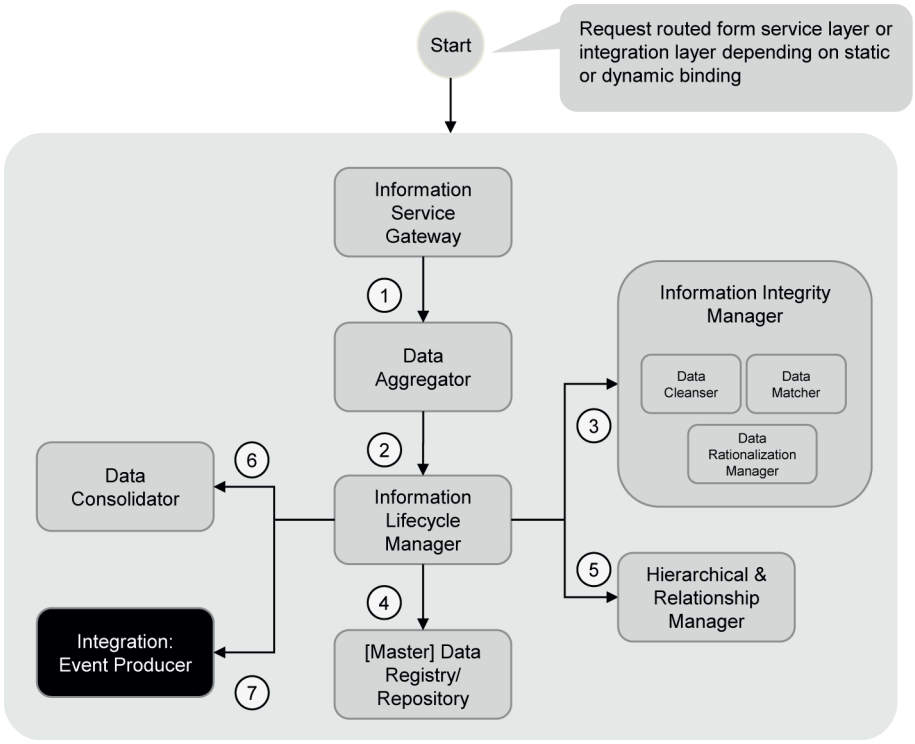


Figure 53 — Key Interactions among ABBs in the Integration Aspect for an Add/Update in an MDM Scenario

In the scenario in Figure 53, the service is to update an MDM the Information Service Gateway uses (2) the Data Aggregator which uses the Information Lifecycle Manager ABB (2) to coordinate the request to update the MDM. First, it gets the data form the Information Integrity Manager (3), which supports cleansing and rationalization of the data to be updated in the MDM. It then updates the MDM (4) with the data and updates the Hierarchical and Relationship Manager (5) so it is in synch with the MDM. Then the Data Consolidator (6) is used to reduce to the data for an event to be sent (7) to record that the MDM has been updated.

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

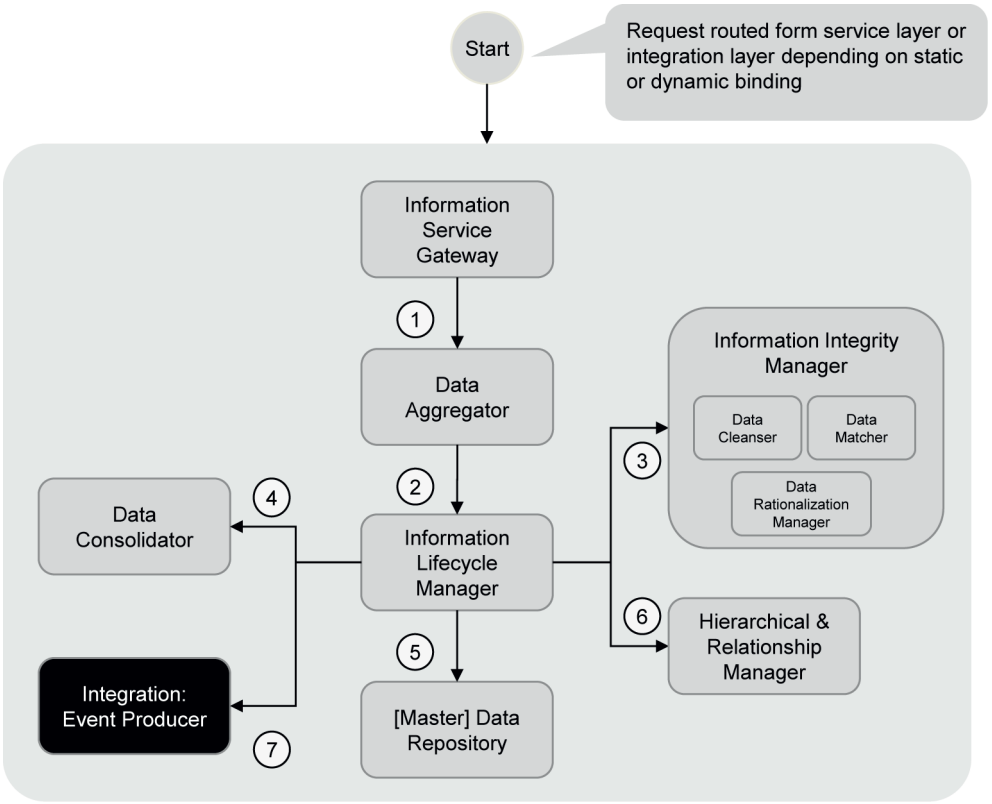


Figure 54 — Key Interactions among ABBs in the Integration Aspect for a Delta Extract and Update in an MDM Scenario

In the scenario in Figure 54, the service is to update an MDM from a delta extract. Like the previous scenario, the Information Service Gateway uses (2) the Data Aggregator which uses the Information Lifecycle Manger ABB (2) to coordinate the request to update the MDM. First, it gets the data form the Information Integrity Manager (3), which supports cleansing and rationalization of the data to be updated in the MDM. However, this time it also gets data from the Data Consolidator (4) which reduces the data before it updates the MDM (5) with the data. Again, it updates the Hierarchical and Relationship Manager (5) so it is in synch with the MDM. An event to be sent (7) to record that the MDM has been updated with the entire extract in the content.

15.4 Significant Intersection Points with other Layers

Certain relationships exist between the ABBs in the Information Aspect with those in other cross-cutting aspects and horizontal layers.

— The Information Service Gateway ABB interacts with the Consumer Layer, Process Layer, Services Layer, Service Component Layer, Operational and IT Systems Layer, Integration Aspect, Management and Security Aspect, and Governance Aspect.

— The Traceability Enabler/Auditor ABB interacts with the Management and Security Aspect.

— The Data Sourcing Manager ABB interacts with the Operational S and IT systems Layer and the Governance Aspect.

15.4.1 Interaction with Cross-Cutting Aspects

The Information Aspect relies on the Cross-Cutting Aspects for the following capabilities. These interactions are based on common scenarios and best practices.

— This layer leverages the description and rules development tools to capture the information models relevant to the service. It also stores the descriptions and rules in the metadata manager and data registry/repository.

— This layer leverages the Event Manager ABB in the Management and Security Aspect for notifying and triggering actions based upon events detected within the data. Events can be defined to support data governance policies, based upon business rules or can be time and date scheduled.

— This layer leverages the Data Transformer ABB in the Integration Aspect for transforming and mapping of data from one format to another and aligning data from different resources.

— This layer leverages the Access Controller ABB in the Management and Security Aspect to enforce security policies and access privileges.

— This layer leverages the Data-Driven Access Controller ABB in the Management and Security Aspect to enforce access privileges on individual data items.

— This layer leverages the Business Activity Monitor ABB, Business Activity Manager ABB, and Activity Correlation Manager ABB in the Management and Security Aspect to monitor events, business activities, Key Performance Indicators (KPIs), to interface with the integration aspect for event notification and propagation, and to analyse the event information, both in real-time/near real-time, as well as stored (warehoused) events, and decide responses to triggered events.

— The Policy Enforcer ABB from the Management and Security Aspect is leveraged by the Governance Aspect to enforce governance policies, by all other layers to enforce security policies, and by the integration aspect to enforce policies during mediation, by the services layer to enforce service policies, and by the process layer to enforce of polices on business processes.

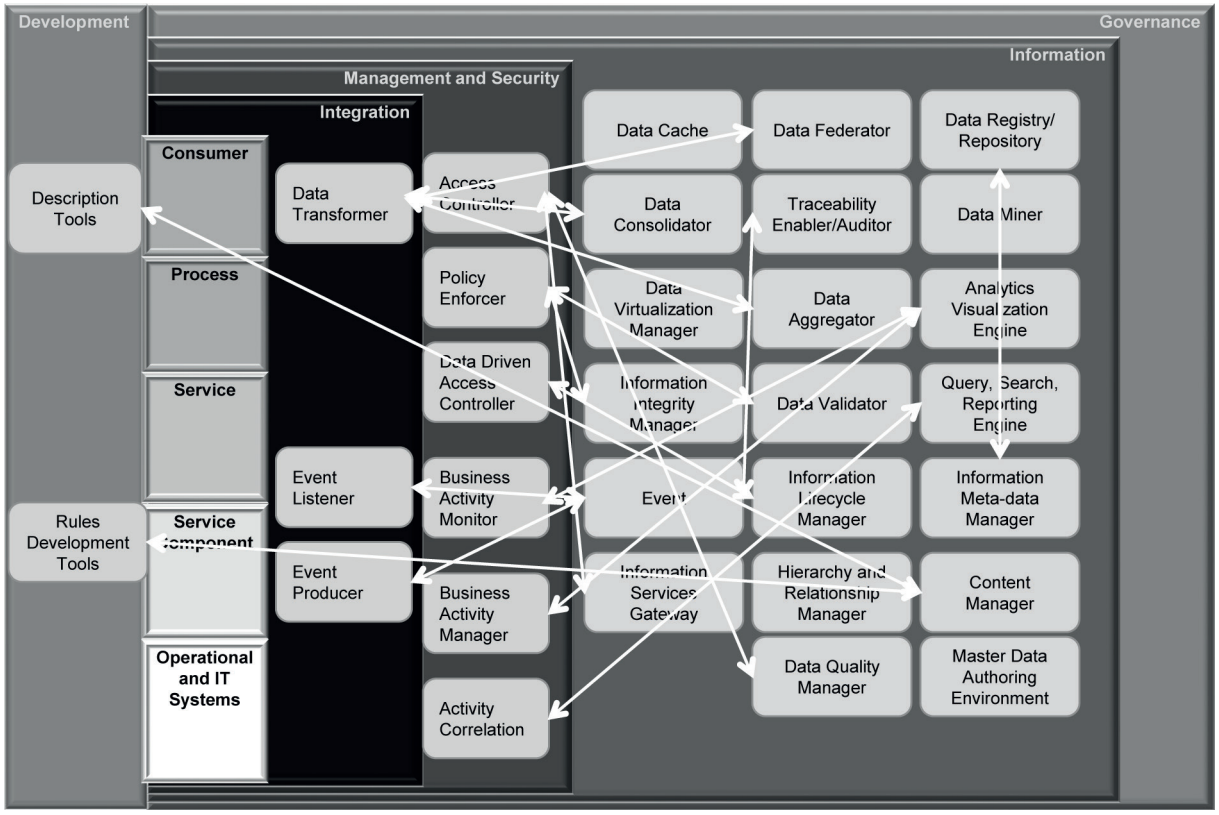


Figure 55 — Key Interactions of the Information Aspect with Cross-Cutting Aspects

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, Process Layer, Services Layer, and Service Component Layer, require information (structure and unstructured data, metadata, and messages) to fulfil their respective responsibilities and therefore rely on the Information Aspect to access information. These horizontal layers are dependent on the ABBs of the Information Aspect to fulfil their information needs. All of the ABBs in the Information Aspect available through the Information services gateway.

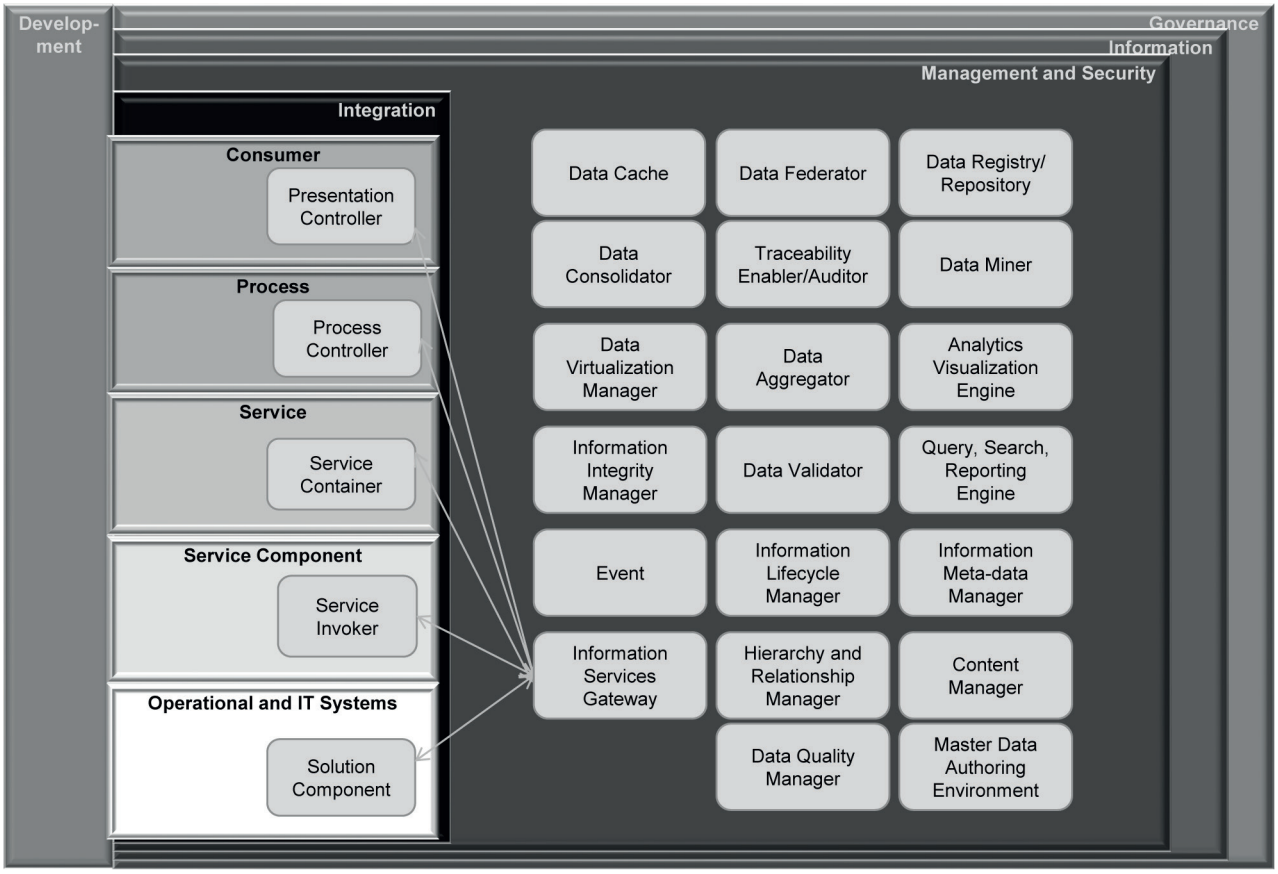


Figure 56 — Key Interactions of the Information Aspect with Horizontal Layers

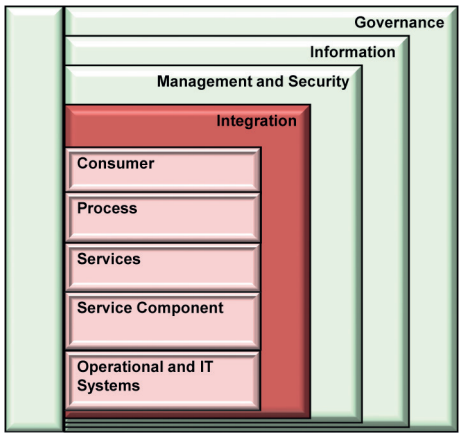
15.5 Usage Implications and Guidance

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

16 Governance Aspect

16.1 Overview

16.1.1 Summary



(From 7.5.10) SOA Governance defines policies, guidelines, standards and processes that reflect the objectives, strategies and regulations to which services and SOA solutions conform. The success of an SOA solution is often in terms of meeting business value objectives. SOA governance activities should conform to Corporate, IT and Enterprise Architecture governance principles and standards relevant to the SOA ecosystem in which the services and SOA solutions are intended to interact. The SOA Governance should also be adapted to match and support an appropriate SOA maturity level.

The Governance Aspect includes both SOA solution governance (governance of processes for policy definition and enforcement), as well as Service governance (service life-cycle). This covers the entire lifecycle and portfolio management of the service and SOA solutions (e.g. SLAs, capacity, performance, security and monitoring). This Aspect also supports governance that needs to be coordinated across organizations, where service consumers and service providers are using services from other organizations.

The goal of the SOA Governance Aspect is to ensure consistency of the service and solution portfolio and supporting lifecycle processes. A given service or SOA solution may be part of more than one portfolio but it conforms to the defined governance for each Corporate or Enterprise domain to which it applies. Thus, some services or SOA solutions may not be appropriate for every governance domain.

The SOA Governance Aspect provides an extensible and flexible SOA governance framework based on ISO/IEC 17998 that supports the alignment of business and IT, including the following:

— Service Level Agreements based on requirements for quality of service and key performance indicators (KPIs);

— capacity and performance management policies;

— design time concerns such as Business Rules.

As a part of the governance framework, a governance regimen (i.e. the customized compliance, dispensation and communication processes to govern the SOA lifecycle and portfolio management) will need to make use of capabilities to store and access governance artefacts, as well as capabilities for managing and enforcing policy, monitoring metrics and managing the configuration and governance of the solution. Organizations may also need a strong change control capability to support changes to governance and the ensuing management of those changes.

To ensure ongoing business and IT alignment, the governance processes and business condition should be continuously evaluated and updated. These separate processes may use the same capabilities as the governance regimen and Management and Security Aspect.

The Governance Aspect supports the following capabilities:

— definition of policies, compliance and exceptions characteristics;

— monitoring the health of SOA services, solution and governance via the Management and Security Aspect;

— identification of metrics reporting on compliance, exceptions, service health, and versions;

— incorporation of for business rules into the governance structure.

16.1.2 Context and Typical Flow

SOA governance helps to ensure that services and SOA solutions are adhering to the policies, guidelines, and standards that are defined as a function of the objectives, strategies, and applied regulations, as well as that the SOA solutions aligned with business needs over time. SOA governance activities should conform to corporate, IT, and enterprise architecture governance principles and standards.

The value of this layer is to ensure that the mechanisms are in place to organize, define, monitor, and implement governance from an enterprise architecture and solution architecture view.

This layer features the following characteristics:

— enables the definition of policies, compliance, and exception characteristics;

— enables monitoring the health of SOA services, solutions, and governance;

— enables reporting on compliance, exceptions, service health, and service versions;

— enables a consolidation point for business rules.

The Governance Aspect is aligned with and makes use of the definitions and best practices described in the standardized SOA governance framework (see Reference [10]). This framework includes definitions and best practices for governance guidelines, roles/responsibilities, governing processes, governed processes, and governance technology and a method for maintaining alignment between business and IT. Other governance standards and governance done using proprietary methods can also be supported by the ABBs of the Governance Aspect.

The governing processes include those dealing with the following.

— Compliance processes ensure that policies requiring conformance are followed. This includes the ongoing vitality processes, such as the Vitality Process described in the SOA Governance Framework, which continuously evaluate the governance processes, key governance model constructs, reference models, and lifecycle (activities, work products) to ensure they continue to be relevant and aligned with the business.

— Dispensation processes and policies to handle exception to compliances.

— Communication processes that disseminate and educate others about governance models.

The Governance Aspect includes 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 the following.

— Service lifecycle management processes – describe the activities, roles for those activities, and work products to the modeling and management of services throughout the lifecycle, from identification to instantiation and retirement. These are often an extension of the software development lifecycle.

— Solution lifecycle management processes – describe the activities, roles that perform them, and work products as they relate to the modeling and management of SOA solutions throughout the lifecycle, from identification to retirement.

— Service portfolio management processes – describe activities for selecting services to be developed and services to be re-used by solutions, ensuring an organization(s) 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 to the needs of the enterprise. Identifying services need by the solutions is one of the responsibilities and ties directly to input to the service portfolio manager.

The SOA Governance Vitality Method phases, Plan/Define/Implement/Monitor, defined with best practices ensure that governance is a long-term process, keeping business and SOA solutions aligned:

— the Plan governance phase is responsible for analysing, arranging, and scheduling solution-level monitoring and management;

— the Define governance phase is responsible for defining a solution-specific SOA-based governance control model and strategy;

— the Implement governance phase is responsible for enabling and realizing solution-level governance control;

— the Monitor governance phase is responsible for monitoring and managing solution-level system status according to predefined governance policies and plans.

Governance ABBs support the various dimensions of governance. This includes the processes needed to create and maintain governance, and, in particular, as it relates to governance vitality and maintaining the governance regimen. Capabilities and ABBs aligned with that standard are defined here.

The SOA Governance Vitality Method phases, Plan, Define, Implement, and Monitor, all require the capability to store and access governance information, define policies with a policy manager, and possibly develop and configure management tools. In addition, the monitor phase needs the ability to monitor metrics, manage and enforce policies, and use change control and configuration management tools to react to policy changes. In addition, workflow can be used to implement the compliance processes.

The governance regimen, i.e. the customized compliance, dispensation, and communication processes to govern the SOA lifecycle and portfolio management, requires capabilities to store and access governance artefacts, 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.

16.1.3 Capabilities

There is a set of categories of capabilities that the Governance Aspect needs to support in the SOA RA. These categories are as follows.

— **Governance Lifecycle**: This category of capabilities provides the ability to plan, define, implement, enable and monitor governance.

— **SOA Metadata Storage and Management**: This category of capabilities enables storing and managing service metadata and governance artefacts.

— **Business Rule Definition and Management**: This category of capabilities provides the ability to define and manage business rules.

— **Policy Definition and Management**: This category of capabilities provides the ability to define and manage policies.

— **Monitoring**: This category of capabilities provides the ability to monitor application of policies, governance processes, and effectiveness of governance.

— **Management**: This category of capabilities provides the ability to manage governance artefacts and processes.

— **Workflow**: This category of capabilities provides the ability to capture and automate governance processes.

This layer features the following capabilities.

— **Governance Planning**

1) Ability to analyse 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 artefacts

— **Governance Enablement and Implementation**

5) Ability to enable governance

6) Ability to realize governance processes

— **SOA Metadata Storage and Management**

7) Ability to store and search any kind of assets and artefacts

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 the following capabilities

   — Service contract definition (e.g. WSDL)

   — service policy management

   — Service version information management

   — Service dependencies (e.g. the ability to integrate with a CMDB tool)

   — Service management descriptions

   — Canonical form and domain model specification for integration with the Information Aspect

10) Ability to store governance artefacts

11) Ability to access governance artefacts

12) Ability to advertise/query for governance artefacts

13) Ability to advertise for services and metadata about services

14) Ability to find or query for services and metadata about services

— **Business Rule Definition and Management**

15) Ability to capture, author, and define business rules

16) Ability to change, manage, and maintain business rules

17) Ability to store business rules

— **Policy Definition and Management**

18) Ability to define policies

19) Ability to correlate business rules into policies

20) Ability to distribute policies

21) Ability to change, manage, monitor, and maintain current policies

— **Monitoring**

22) Ability to monitor solution-level system status according to predefined governance policies and plans

23) Ability to manage solution-level system status according to predefined governance policies and plans

24) Ability to measure and gather metrics on the SOA services, SOA solutions, governing processes, and policy enforcement

25) Ability to evaluate metrics and test against policy regularly

26) Ability to use metrics to determine appropriateness of the current governance regimen

27) Ability to trigger checkpoints in the compliance process

28) Ability to indicate the status of governing and governed processes and their metrics in real-time

29) Ability to report the results of governing processes and their effectiveness

— **Management**

30) Ability to do configuration management to implement and maintain governance

31) Ability to do change control to implement and maintain governance

32) Ability to access control and apply security policies for governance processes

— **Workflow**

33) Ability to capture governing processes as workflow documents

34) Ability to automate governing processes

16.1.4 Structural Overview of the Layer

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

— ability to store and access (i.e. with registries, repositories, websites, or databases) artefacts related to governance (artefacts can be organization documentation, business process documentation, policies, compliance records, etc.),

— ability to define and manage business rules,

— ability to authorize and manage policies based on business rules at all phases of the SOA solution (design and runtime), including the ongoing governance vitality phase,

— ability to measure, monitor, and access governance metrics, both for ensuring governance policies are adhered to and for ensuring governance regimens continue to be appropriate,

— ability to manage and maintain governance, including the ability to do configuration, change control, security, and reporting, and

— ability to automate processes and capture processes in workflows.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 16.1.3 that the ABB supports. For example, in Figure 57, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Development Aspect is shown as light grey (with a prefix of "Development"). For example, in Figure 57, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. MaS: Access Controller supports capability number 32: '32 ability to access control and apply security policies for governance processes.' The Development: Development Tools supports capabilities 3, 4, 5, and 6. 6 is: '6: Ability to realize governance processes'.

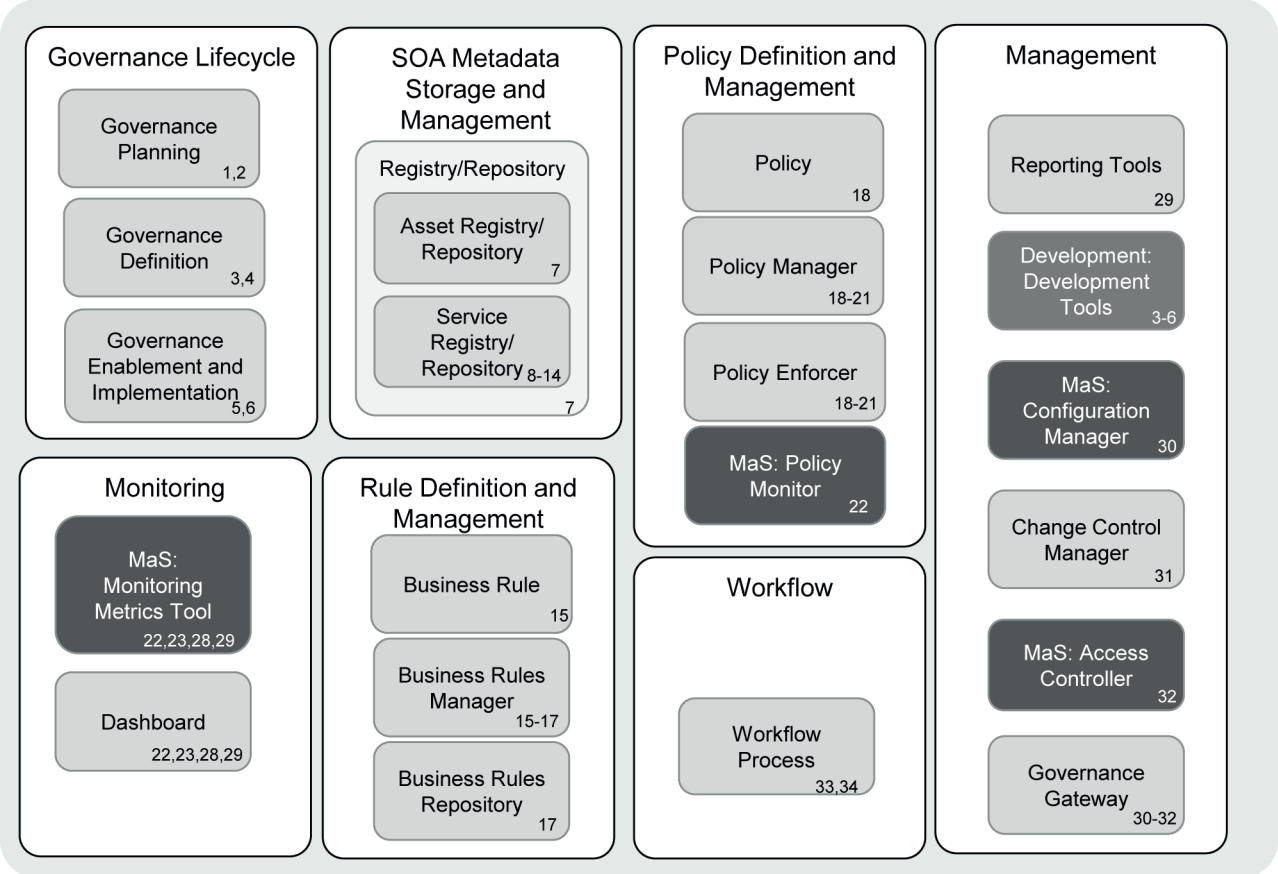


Figure 57 — ABBs in the Governance Aspect

The Governance Aspect leverages ABBs from the Management and Security Aspect to fulfil its core responsibilities. The ABBs from the Management and Security Aspect leveraged by the Governance Aspect are Policy Monitor ABB, Monitoring Metrics Tool ABB, Configuration Manager ABB, and Access Controller ABB.

SOA governance capabilities and ABBs are used during the governing of an SOA solution. Other layers in the SOA RA define what technologies should be used to develop, deploy, and operate an SOA solution. The same technologies can be used to support the SOA solution and governance of the SOA solution.

Technology used to realize 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 may use a different set of ABBs.

— A Plan Governance phase is responsible for analysing, arranging, and scheduling solution-level monitoring and management; therefore, it would use the Service Registry/Repository ABB, Asset Registry/Repository ABB, and Business Rules Registry/Repository ABB to store governance information artefacts like governance principles, organization roles and responsibility, and business rules. The Registry/Repositories introduced here represent logical entities that may be implemented as separate repositories, a federation of such Registry/Repositories, or as a consolidated registry/repository.

— A Define Governance is responsible for defining a solution-specific SOA-based governance control model and strategy; therefore, it would use the Service Registry/Repository ABB and Asset Registry/Repository ABB to store the information artefacts, governance processes, and transition processes for implementing governance. It may also use a Business Rules Manager ABB and Policy Manager ABB to author policies to be used in the implementation phase.

— An Implement Governance phase is responsible for enabling and realizing solution-level governance control; therefore, it would use many of the governance ABBs, including the Service Registry/Repository ABB for governance information for services, Service Registry/ Repository ABB for services and governance metadata, Policy Manager ABB to author policies, and the Management and Security Aspect: Configuration Manager to configure the Management and Security Aspect: Policy Enforcer ABB; Management and Security Aspect: 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 Management and Security Aspect: Policy Enforcer ABB, Management and Security Aspect: Monitoring Metrics Tool ABB, Dashboard ABB, and Reporting Tools ABB.

Each of the governing processes in the final governance regimen, compliance, dispensation, and communication may use a different set of ABBs. An example compliance process is illustrated in 16.2.

16.2 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 registry/repository or a policy enforcement tool).

SOA governance ABBs represent 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.

The details of the ABBs are grouped by the capabilities.

16.2 Supported Capabilities

16.2.1 Governance Lifecycle

16.2.1.1 Governance Planning

This ABB represents the planning processes for analysing existing governance, identifying goals, strategies, principles and roles and then arranging and scheduling solution-level monitoring and management.

16.2.1.2 Governance Definition

This ABB represents the process of defining a solution-specific SOA-based governance processes, artefacts, control model and strategy.

16.2.1.3 Governance Enablement and Implementation

This ABB represents the processes used to implement governance processes, as well as enabling and realizing solution-level governance control. It also monitors and manages solution-level system status according to predefined governance policies and plans.

16.2.2 SOA Metadata Storage and Management

16.2.2.1 Registry/Repository

This ABB represents a generic storage and enables organizations to organize, govern, and manage their artefacts and assets scattered throughout the enterprise and to encourage re-use of existing assets. It provides the ability to search for artefacts and assets by different perspectives such as for general description, classification, usage, and content. The registry/repositories introduced here represent logical entities that may be implemented as separate registry/repositories, a federation of such registry/repositories, or as a consolidated registry/repository. This ABB is used at design time and runtime.

16.2.2.2 Asset Registry/Repository

This ABB represents a specialization of the general Registry/Repository ABB that organizes, governs, and manages assets encouraging re-use of assets. Assets could be business processes model, service artefacts, components design and code, models, documents, etc. Standards for asset registry/repository and management include Re-usable Asset Specification (RAS) from OMG.

16.2.2.3 Service Registry/Repository

The ABB represents a specialization of the general registry/repository ABB that enables advertising and discovery of available services and supports the runtime binding of services and service virtualization. This ABB stores and locates metadata about services, including descriptions of the service contract, information about the QoS policies and security, versioning information, and runtime information such as endpoints. This ABB integrates with the Service Performance Manager ABB to support the runtime information collection and storage in order for users to evaluate service performance.

It is responsible for storing and accessing governance artefacts and best practices for SOA solutions and governance of SOA solutions from various perspectives, including organizational concerns, development concerns, runtime operational concerns, and lifecycle management concerns. Artefacts stored to guide governance may include service registrations, policy, guidelines, and governance processes. Services and metadata are available to the solution, as well as governance processes. Advertisement of services should be governed.

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/repository). The other concern is the administration of the service where there are changes to the location of a service. Location is expressed in terms of an "end-point"; a.k.a, the location from which a service is invoked. This could be the service container's address or some other unique identifier (URI). Standards for registries include UDDI.

16.2.3 Rule Definition and Management

16.2.3.1 Business Rule

This ABB represents a business rule constraining some concerns of business such as business processes, services, and information. Business rule is one of the fundamental constructs of an SOA solution and analysis and design based on the service oriented paradigm. The business rules may apply throughout the SOA solution and governance lifecycle.

16.2.3.2 Business Rules Manager

This ABB represents the capabilities 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.

16.2.3.3 Business Rules Registry/Repository

This ABB represents capabilities for storing and accessing business rules artefacts. This ABB is used by the Business Rule Manager ABB.

16.2.4 Policy Definition and Management

16.2.4.1 Policy

This ABB represents a policy describing principles to guide decisions to drive to desired outcomes. Policies may apply throughout the SOA solution and governance lifecycle. Policies could be at multiple levels, such as business level, architectural level, and/or operational level. It is important to capture the policy-related decisions and rules and policy information and its sources while defining policies such that the policies can be evaluated during policy enforcement by the Policy Enforcer in the Management and Security Aspect.

16.2.4.2 Management and Security Aspect: Policy Enforcer

See 14.2.7.1.

16.2.4.3 Management and Security Aspect: Policy Monitor

See 14.2.7.2.

16.2.4.4 Policy Manager

This ABB represents capabilities for defining, authoring, distributing, and maintaining policies using policy management tools. Sophisticated policy managers may also check for and resolve policy conflicts. This ABB represents the primary Policy Administration Point in the SOA RA. This ABB is responsible for the distribution of the policies to one or more Policy Decision Point represented by the Policy Enforcer ABB in the Management and Security Aspect and its intersection with the other layers of the SOA RA for evaluation and enforcement purposes.

It is important to note that this ABB supports the management of policies needed to support security and logically includes all the responsibilities of a security policy manager.

16.2.5 Monitoring

16.2.5.1 Management and Security Aspect: Monitoring Metric Tools

See 14.2.4.9.

16.2.5.2 Dashboard

This ABB represents a set of tools that provide real-time status of the governing and governed processes and their metrics and checkpoints. It leverages the Monitoring Metric Tools ABB and Policy Enforcer ABB in the Management and Security Aspect.

16.2.6 Management

16.2.6.1 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 Registry/Repository ABB. These reports can be created during any governance phase and any of the governance processes.

16.2.6.2 Development Aspect: Development Tools

See 17.2.3.11.

16.2.6.3 Management and Security Aspect: Configuration Manager

See 14.2.8.1.

16.2.6.4 Change Control Manager

This ABB represents capabilities 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.

16.2.6.5 Management and Security Aspect: Access Controller

See 14.2.2.8.

16.2.6.6 Governance Gateway

This ABB represents the gateway of all the ABBs in the Governance Aspect 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.

16.2.7 Workflow

16.2.7.1 Workflow Process

This ABB represents capabilities that enable the automation of compliance and dispensation processes. The Workflow ABB from the Process Layer enables a business process to support manual intervention. This is often a requirement in a situation where error handling has to be performed. Standards include BPEL.

16.3 Inter-Relationships between the ABBs

Figure 58 illustrates the different ABBs and their inter-dependencies.

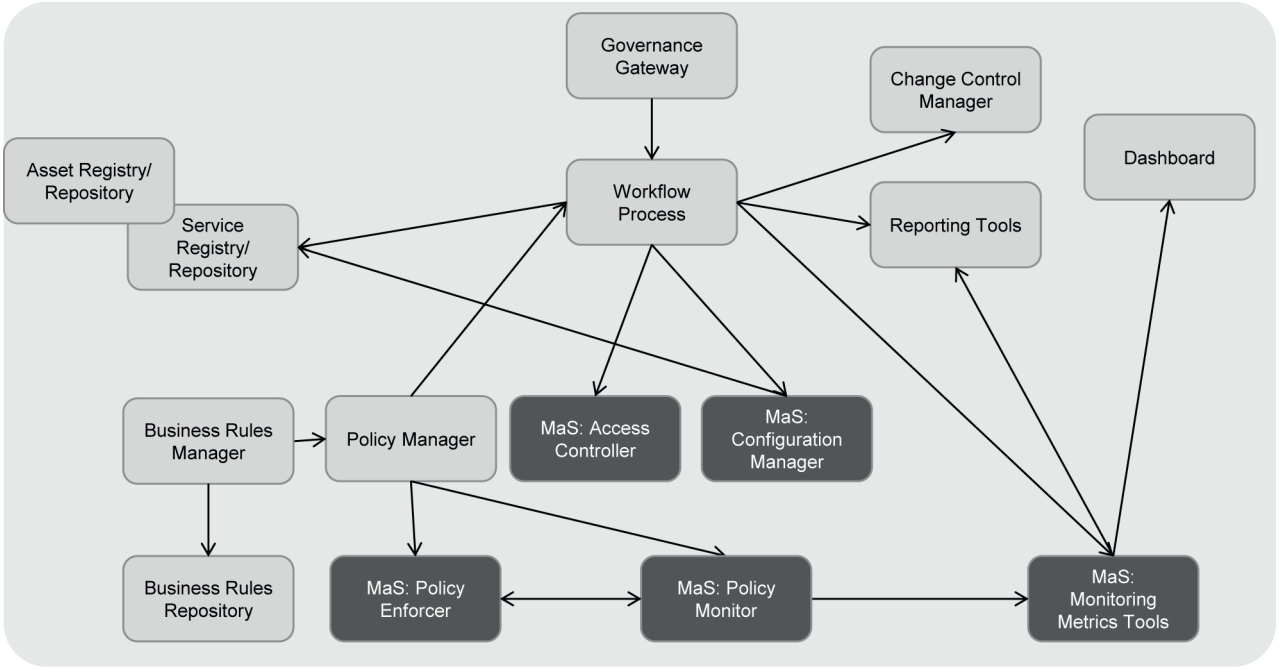


Figure 58 — Relationships among ABBs in the Governance Aspect

The Governance Gateway ABB invokes and governs the governing processes captured as workflows in the Workflow Process ABB.

The Workflow ABB captures and documents the governing processes. The workflows find governed services and other services to support governance using the Service Registry/Repository ABB. The Workflow ABB interacts with the Access Controller ABB, Policy Enforcer ABB, and Configuration Manager ABB in the Management and Security Aspect and Change Control Manager ABB and Reporting Tools ABB in the Governance Aspect 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 Management and Security Aspect 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 Registry/Repository ABB or Service Registry/Repository ABB, Policy Enforcer ABB in the Management and Security Aspect, Access Controller ABB in the Management and Security Aspect, and governing process workflows.

The Policy Enforcer ABB in the Management and Security Aspect and governing process Workflow ABB shares policy results with the Monitoring Metrics Tool ABB in the Management and Security Aspect.

Figure 59 shows a flow for an example compliance process.

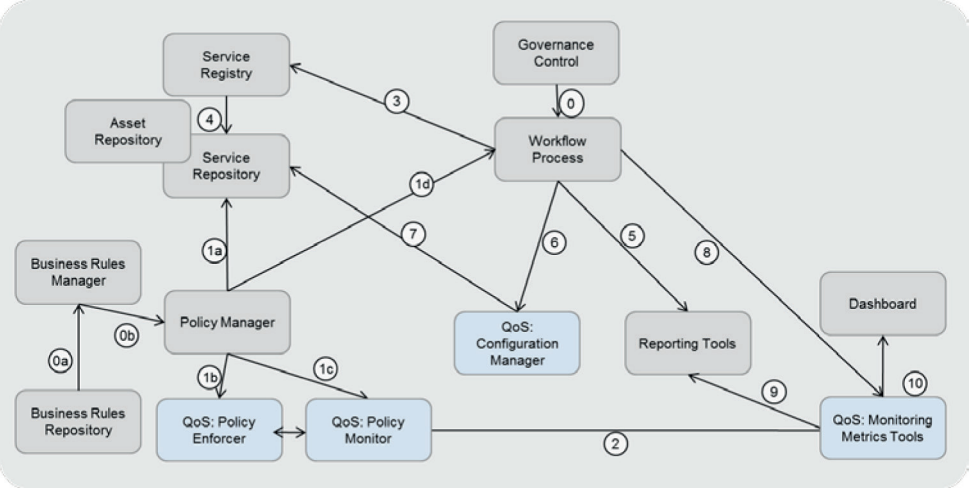


Figure 59 — Sample interactions among ABBs in the Governance Aspect for a Governance Compliance Process

In this example, a business rule has indicated that a policy is set that services that have excessive failures shall be inactivated and operations notified via the Dashboard ABB.

a) In 0a-0b, the business rules are distributed from the Business Rules Registry/Repository to the Business Rules Manager who in turn sends them to the Policy Manager. The business rules are established by the business using Governance Control (0) and translated into policies by the Policy Manager. These policies are consulted throughout the scenario.

b) At (1a-d), the policy to inactivate a service after five failures in a day is defined and distributed to the service registry/repository, policy monitor, policy enforcer, and compliance process workflow. When the policy monitor detects that the service has failed more than five times, it invokes the policy enforcer to inactivate the service and notifies (2) the monitoring metrics tool.

c) The compliance process is running and at (3) looks up the service information which retrieves the policies for the service from the registry/repository using (4).

d) Now, the compliance process checks with the monitoring metrics for policy exceptions using (2) and finds that the service has exceeded its failure threshold.

e) The compliance process interacts with the configuration manager using (6) to configure the service to be out-of-use, the configuration manager interacts with the status manager is the Management and Security Aspect to update the registry/repository with the current service configuration set to out-of-use using (7).

f) Now, the workflow reports that the service has been taken out of use using (5), as does the monitoring metrics.

g) The monitoring metrics update using (10) the dashboard with a new status for the service being out of use.

Here, it is clear that the Governance and Management and Security Aspects work cooperatively.

16.4 Significant Intersection Points with other Layers

The Governance Aspect 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 Aspect. Some of the layers, especially Quality of Service, Integration, Services, and Process layers contain ABBs that the Governance Aspect needs to leverage.

16.4.1 Interaction with Cross-Cutting Aspects

The Governance aspect provides capabilities necessary to enable for governance for all the other cross-cutting aspects, namely, Integration, Information Architecture, and Management and Security Aspects. In addition, the Governance Aspect relies on ABBs in the Management and Security Aspect to fulfil its core responsibilities. These interactions are based on common scenarios and best practices.

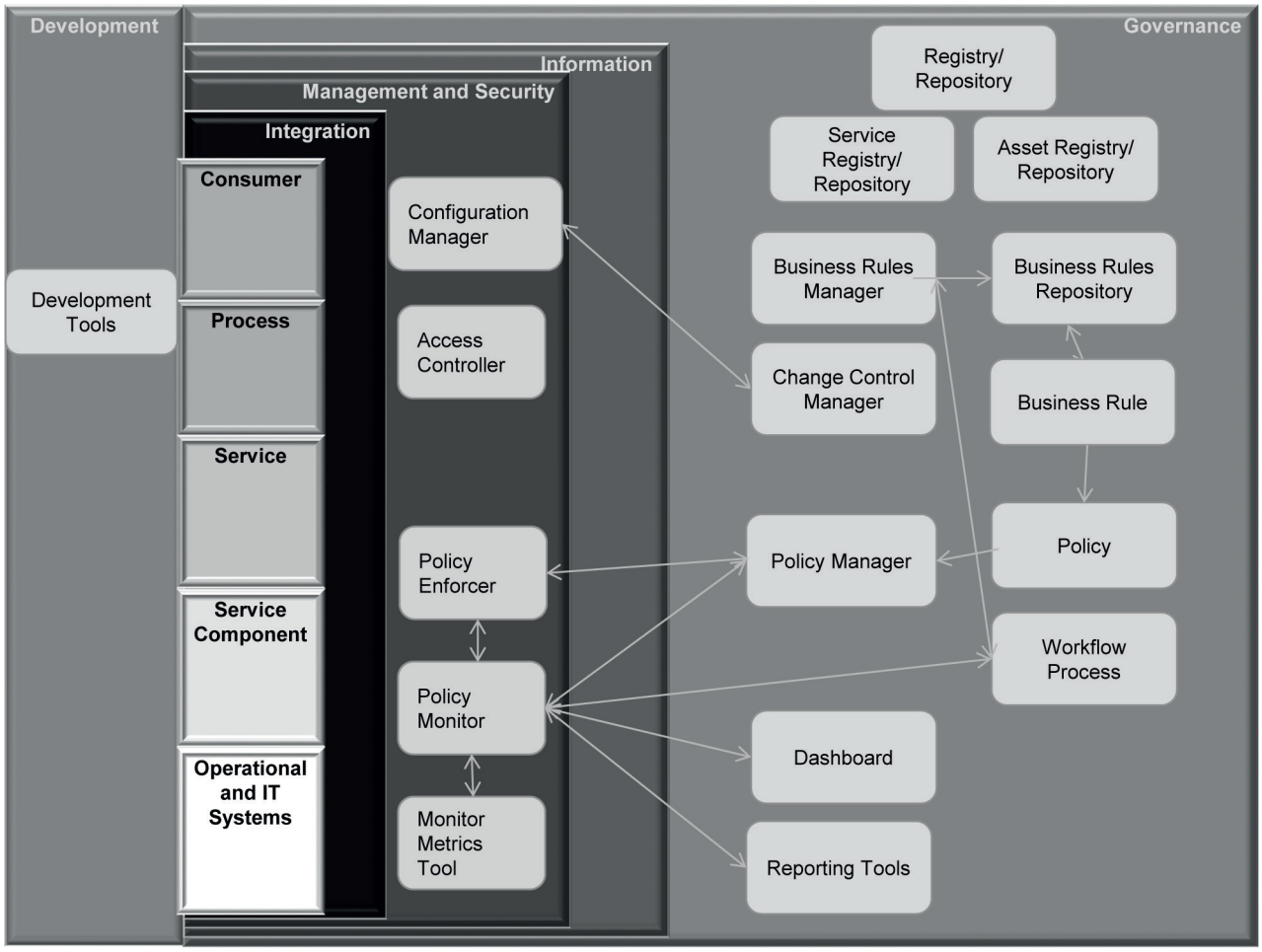


Figure 60 — Key interactions of the Governance Aspect with Cross-Cutting Aspects

Governance drives the definition of policies used to drive the concerns of QoS in the Management and Security Aspect. The Governance Aspect is dependent on the Management and Security Aspect to provide the following capabilities.

— The Development Aspect is used to create descriptions and rules that are managed by the business rule manager and used to create policies that are stored by the policy manager. The capability for controlling change uses the maintenance and versioning tools of the Development Aspect.

— The Business Rule Manager ABB is needed to ensure that the appropriate policies are set to support the policy manager capability. It also supports the Policy Enforcer ABB in the Management and Security aspect in fulfilling its responsibilities through the Policy Manager ABB.

— It leverages the Policy Enforcer ABB in the Management and Security Aspect to enforce policies related to change control processes as required by the Change Control Manager ABB to ensure that change control is performed appropriately. Similarly, it leverages the Policy Enforcer ABB in the Management and Security Aspect to enforce security policies and policy to configure the solution using the Configuration Manager ABB in the Management and Security Aspect. It also leverages the Policy Enforcer ABB in the Management and Security Aspect to enforce governance policy and monitoring metrics for a solution.

— It leverages the Access Controller ABB in the Governance Aspect to define the policies used to configure the security for the SOA solution and the governing processes via the security manager capability.

— It leverages the Configuration Manager ABB in the Management and Security Aspect in solution configuration and changing governance workflow processes. The Governance Aspect defines the policies used to configure the solution using the Configuration Manager ABB in the Management and Security Aspect. In case there are automated governing process workflows, the workflow adjusts and changes the configuration using the Configuration Manager ABB in the Management and Security Aspect in order to adhere to the governance policies.

— It leverages the Monitoring Metrics Tools ABB and Policy Enforcer ABB in the Management and Security Aspect to measure, gather, evaluate, and test metrics against policies on a regular basis. The Governance Aspect defines the policies used to implement the monitoring of metrics of the Governance Aspect and SOA solution. Metric monitoring and analysis is used to drive governing processes and workflows to correct any policy violations and use the dashboard. Metrics and policy exceptions are also used to drive re-evaluation of the current governance regimen. Metrics are gathered on SOA services, governed processes, and governing processes. The dashboard ABB leverages the Monitoring Metrics Tools ABB in Management and Security Aspect to customize what metrics and events should be visible on the governance and solution dashboard.

16.4.2 Interaction with Horizontal Layers

Governed assets exist in all of the horizontal layers, for example:

— IT infrastructure and implementation assets are governed in the Operational and IT Systems Layer;

— enterprise component are governed in the Service Component layer;

— services are governed in the Services Layer. Policies defined by governance decide which services will be built and re-used;

— business processes are governed in the Process Layer;

— governance and integration intersection, policies defined by governance, govern the mediation of interactions. The Integration Aspect will utilize the Registry/Repository to actually find an end-point as a result of service invocation.

These horizontal layers define the different kinds of policies by interfacing with the Policy Manager ABB. In addition to governing these horizontal layers, the Governance Aspect uses the Process Layer to capture governance process and the Workflow ABB leverages the Process Layer to define the governance processes.

The Services Layer leverages the Service Registry/Repository ABB to store service definition/contracts, policy, and metadata about services during design time. The Services Layer leverages the Service Registry/Repository ABB to store service definition/contracts and policy and metadata about services to be used during runtime in order to discover services and bind to service provider/end-point enabling service virtualization.

The Integration Aspect leverages the Service Registry/Repository ABB to determine the end-point for a service request and to enable service virtualization.

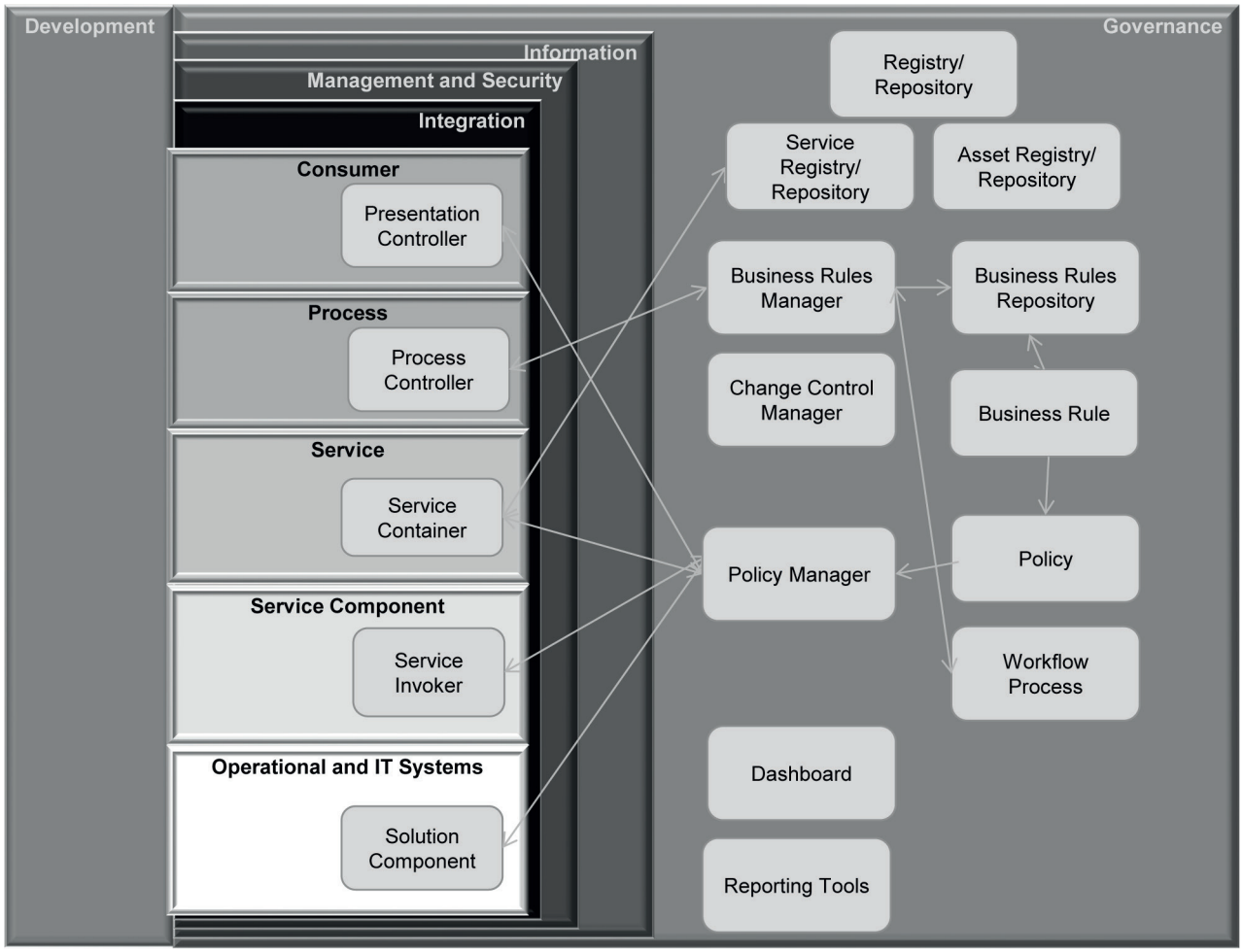


Figure 61 — Key Interactions of the Governance Aspect with Horizontal Layers

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

— The Service Registry/Repository ABB is used by the Services, Business Process, Consumer, Integration, and Management and Security Aspects.

— The Policy Manager ABB is used by the Integration and Management and Security Aspects.

— The Business Rule Manager ABB is used by the Business Process, Services, Service Component, and Management and Security Aspects.

16.5 Usage Implications and Guidance

At the heart of these processes is the Service Model, the unifying concept that binds these elements to together and makes them relevant.

16.5.1 Options and Design Decisions

Four of the design decision points that exist are

— the use of a standard service registry/repository *versus* roll-your-own,

— collaboration technologies for communication and vitality,

— automation of service lifecycle and tracking, and

— automation of compliance and exception handling processes.

The information in this layer that is collected and made available via a registry/repository consists of, for example,

— guidelines for SOA governance,

— guidelines for service and SOA solution lifecycle and portfolio management,

— best practices,

— business rules,

— policies (e.g. security),

— standards,

— service and SOA solution roadmaps, and

— compliance, dispensation, and communication documentation.

As a result, it is necessary that the Governance Aspect capabilities and ABBs support all of governing all of these processes. The governance of each of these processes may require different ABBs.

Another important responsibility of the Governance Aspect is to measure, gather, evaluate, and test metrics against policies on a regular basis. KPIs for this layer may include

— usage metrics of a service,

— downtime and failure statistics on a service or service set,

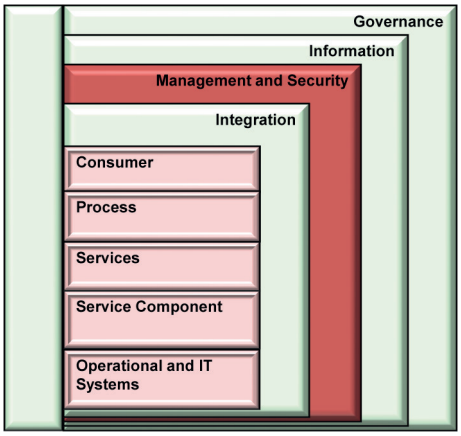
— policy violations, and

— number of services compliant and number applied for exceptions.

17 Development Aspect

17.1 Overview

17.1.1 Summary



(From 7.5.11) The Development Aspect contains all of the components and products needed to develop and change implementations of SOA services and solutions. The service implementations should include the development of or use of implementations in the Operational Systems and IT layer, Service Component Layer, Service Layer, Process Layer and cross-cutting Aspects. Service implementations should encapsulate existing systems and resources such that late binding of services can be supported to promote loose coupling.

Development includes solution and service design, modeling, implementation and deployment. Operational and management capabilities are the responsibility of the Management and Security Aspect. Maintenance uses capabilities from Development Aspect and Management and Security Aspect.

Tools that support the Development Aspect include the entire suite of architecture tools, modeling tools, development tools, visual composition tools, assembly tools, methodologies, debugging aids, instrumentation tools, asset repositories, source code control, discovery agents, and publishing mechanisms that may be used to construct a SOA solution.

The Development Aspect supports the following capabilities:

— development, configuration, debugging and testing environments for the construction of services;

— testing of services and SOA solutions, ranging from isolated testing to testing within the operational environment or ecosystem;

— coordination with monitoring to effectively provide for continuous testing throughout the operational lifetime;

— service encapsulation of existing application systems or data resources;

— reusing existing assets to develop services.

17.1.2 Context and Typical Flow

The Development Aspect defines the capabilities and ABBs needed to develop and support the implementation of services. There are three elements required in a Services Oriented Architecture (SOA), the services themselves, consumers of those services, and the agreement between the providers and the consumers that make it possible for the services to be consumed. The agreement is established through descriptions, specifications, and standards. The building blocks consist of programming languages and tools needed to describe and create realizations of the services and allow their consumption by consumers.

An addition, the Development aspect provides support for deployment and publication, continuous testing, and maintenance of the services. This layer supports the clear separation of concerns between service development and operations. This separation supports SOA and cloud deployments where the operations are likely independent of the development of the service but also allows the common situation where both development and operations are conducted by the same organization.

17.1.2.1 Roles

Capabilities and ABBs are used by one or more roles that participate in the lifecycle of a SOA solution. For this SOA RA, those roles are as follows.

— Service Provider – provides and manages operational services.

— Consumer – uses services via invoking them with messages that are compliant with the service definition.

— Service designer; Service Architect – designs the higher level structures that are needed as input into the development of a service and its related and dependent elements.

— Service Description Developer – develops descriptions of services which include service interfaces (like WSDL), policies, service level agreements and other definitions for a service or SOA solution. Service descriptions help ensure that whoever is providing and consuming are interoperable. The Service Description developer may store or publish the service descriptions in a registry/repository.

— Consumer Implementation Developer – develops implementations of the solution that consumes services and user interfaces to the services. The Consumer implementation developer gets requirements and service descriptions and then may create implementations using ‘skeletons’ or

stubs generated by tools from the descriptions. Consumer implementation developer may test the services and therefore may need the ability to execute deployment.

— Provider Implementation Developer – develops the implementation of the service, service components, appropriate components from the Operational and IT Systems Layer and necessary cross-cutting aspects in conformance with the service description. The provider implementation developer gets requirements and service descriptions and then may create the service implementation. This role may test and then may hand over the service implement to the deployment engineer for deployment. When developers create services, like all services, they may use or expose other services or processes.

— Test Engineer – tests the services or updates to services during implementation development and before production deployment by the service provider.

— Deployment Engineer – designs and develops the deployment packaging, artefacts and processes for the service and solution implementation that the service provider's deployment engineers will use to deploy the service.

In 17.1.2.3, the capabilities and ABBs needed by each of these roles will be articulated.

17.1.2.2 Assumptions

— A service contract is the definition of the service syntax, semantic description of the function of the service and agreement on the quality of service concerns, including management, security and service levels.

— If a change impacts how the service is consumed, then the service needs a new service version.

— Services should be reusable. Developers on consumer and provider side are working against the same service description (usually attached to a contract that they both agree to).

— Developers, providers and consumers can be in the same organization or different organizations.

17.1.2.3 Flow of artefacts and work between roles

— Service Development involves the implementation of a service and creation of deployment packaging in a way that allows the deployment and operation of the service in the execution environment.

— The Service Description Developer creates a description of the service that defines a contract that both the Service Provider and Consumer follow to insure interoperability of the service.

— The Service Provider and Consumer adopt the same service contract. The Service Implementation Developer creates the implementation of the services that adhere to the service contract. The Service Implementation Developer also develops packaging of the service implementation to enable it to be deployed by the Service Provider.

— The Service Consumer Developer creates the implementation needed to invoke the service in a manner consistent with the contract. The Service Consumer Developer may need to integrate with other systems to process the data from the service.

17.1.2.4 Service lifecycle

Understanding the service lifecycle is important for understanding the roles, capabilities and ABBs defined by the Development Aspect. There are many to choose from that can be used here but to use a very simple one with four stages.

— Model. In the model phase, requirements for the service and SOA solution are gathered and prioritized with business requirements. The service and solution are modelled and designed. From this phase, there is produced a set of specifications and descriptions for the services and the solutions.

— Assemble. In the assemble phase, the services are implemented, including implementing using compositions of other services and processes. Existing resources may need to be discovered or acquired and then composed into the solution. Testing of the assembled solution is a key task.

— Deploy. Deploy phase integrates people, processes and information. The assembled implementation is evaluated and then deployed and provisioned into production.

— Manage. Operate the services and solutions, including monitoring and reporting on metrics and business metrics on services, applications, identity and compliance. Managing includes deprecating, and eventually retiring, a deployed service or solution, when it is no longer required.

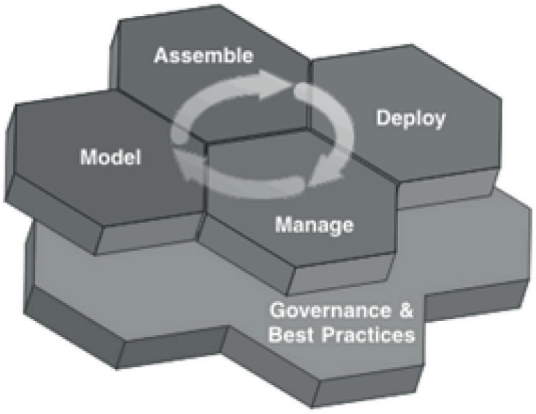


Figure 62 — SOA Lifecycle and Governance

There are roles that participate in each of these phases (see Table 1).

— The Service Description Developer participates in the Model Phase creating descriptions.

— The Service Implementation Developer participates in the Assemble Phase creating service and solution implementations.

— The Test Engineer participates in the Assemble Phase testing the service and solution.

— The Deployment Engineer participates in the Deploy Phase deploying the service and solution into the Service Providers platform.

— The Service Provider participates in the Manage phase operating the service at runtime.

— The Service Provider may participate in all phases: in the Model, assemble Deploy and Manage phases but has to participate in the Deploy and Manage phases.

Table 1 shows which lifecycle phases each of the roles is involved in.

Table 1 — Roles to Lifecycle Phases Mapping for the Development Aspect

Role	Model	Assemble	Deploy	Manage
Service Provider	x	x	x	x
Service Description Developer	x			
Service Architect	x			
Service Provider Implementation Developer		x		

Table 1 *(continued)*

Role	Model	Assemble	Deploy	Manage
Service Consumer Implementation Developer		x		
Service Designer	x			
Test Engineer		x		
Deployment Engineer			x	
Service Consumer	x	x		x

17.1.2.5 describes the description artefacts to support the activities for the roles. For services to be offered and used, three essential concerns need to be described and available to the consumer, as follows:

— syntax - how the data is represented;

— semantics - what the service does - (e.g. 'retrieves customer profile' or 'return payment schedule');

— quality of service – policies for using the service and agreements on performance.

A set of artefacts has been defined to capture those descriptions for the consumer, as well as the descriptions needed to enable a service developer to create an implementation of the service that will run in the target runtime execution environment. These artefacts are as follows.

17.1.2.5 Description Artefacts

— **Service Contract** – the specification of the service that a consumer uses to interact with the service. It includes the definition of the service syntax, semantic description of the function of the service and agreements on the quality of service concerns, including security and service levels. It may include other description artefacts: the service interface, the service level agreements, and policies. This specification is agreed to and adhered to by the provider and consumer. The other description artefacts are often referred to by or contained in the contract.

— **Service Interface** - the definition of service inputs and outputs and a method of invocation. It is the syntax (how the data is represented) for the contract. It includes event formats for any events the service may include. It can define the following:

  — method of invocation;

  — error responses or fault conditions;

  — optional element specification patterns (element defined to be nil vs no element defined may have different meanings but is exactly the same on the wire).

— **Service Level Agreement (SLA)** – the definition of the quality of service (QoS) of the interactions and performance between the service provider and service consumer. It is provided by the service provider to the service consumer as part of the terms and conditions for using the service. It can define any one of, but not limited to, the following:

  — security constraints;

  — reliable messaging requirements;

  — transactional requirements;

  — common metrics for availability, scalability and performance guarantees, for example, availability time, response time, etc.;

  — versions of service;

  - policies that affect the consumers use of the service such as disaster and recovery, high availability, accessibility.

- **Policy** – the set of policies associated with the service that the service provider has defined for the service that regulates the lifecycle, operational management, and governance of the service. The consumer of the service may need access to policies associated with their service level agreement. Optional.

- **Service Deployment Descriptor** – the description of the service components and information needed by the service deployment engineer to take a set of artefacts, including implementations and descriptions, and deploy the service implementation into an Operational and IT Systems Layer or platform and expose the service so it can be used by the consumers. This implies that a mechanism exists so that consumers can find or discover the service and that mechanism is aware of the service and how the service is designed to be accessed. Optional.

- **Hosting Environment Descriptor** – the description of the requirements of the service implementation. The target runtime execution environment and intended operational management of the service may have an impact on the implementation; those impacts and expectations need to be described for the service implementation developer. This may include enablement and instrumentation of metrics and operations for availability and performance monitoring, management, auditing, logging, and configuration. Optional.

- **Process Description** – the description of the processes that need to be or have been used to implement the services. Optional.

- **Event** – the description of the event formats that the service may issue or is expecting to receive. Sometimes, these are included in the service interface specification when the consumers are impacted by the events. Optional.

Table 2 shows when artefacts are crated and used in the lifecycle.

Table 2 — Artefact to Lifecycle Phase Mapping for the Development Aspect

Description artefacts	Model	Assemble	Deploy	Manage
Service Contract	x	x		
Service Interface	x	x		
Deployment Descriptor		x	x	
Policy	x	x	x	x
Service Level Agreement			x	x
Hosting Environment Descriptor	x	x		
Process Description	x	x		
Event	x	x		

17.1.2.6 Business Process Implementation

Business process implementation for the service provider involves the implementation of workflows with or without human interactions and the support for long-running transactions. The Development Aspect capabilities need to support

a) the specification and storage of orchestrations, using standards such as BPEL, and

b) the specification and storage of workflows involving standards such as BPMN.

The Development Aspect supports the separation of roles in the Cloud computing ecosystems currently emerging where the developer of services and the service providers are different companies.

There are examples of the specification of the Service Contract and the Binding of the service contract to the underlying Service semantics in web services and REST. For web services, and in the context of SOAP-style technologies, WSDL specifications, associated schemas and their associated binding are defined. For REST, in the context of RESTful services, WADL, associated schemas and their associated bindings are defined. This capability of contract specification and binding of the underlying semantic implementation applies to both traditional, internal SOA services, as well as to Cloud, web, and infrastructure services.

In a SOAP environment, WSDL and Schema Annotations can be leveraged to describe semantics specifications but textual descriptions are very common as well.

17.1.3 Capabilities

There are multiple categories of capabilities that the Development Layer needs to support. These categories of capabilities are as follows.

— **Description Development** – ability to document descriptions of services and contracts.

— **Implementation Development** – ability to develop service implementations for both the provider and consumer.

— **Operations Enablement** – ability to enable the service implementation to be managed and governed by the service provider.

— **Publication** – ability to discover and publish services.

— **Testing** – ability to test services functionally and for interoperability.

— **Maintenance/Changes** – ability to fix, maintain and improve the service.

— **Process Development** – ability to define and develop processes as part of a service implementation.

— **Deployment** – ability to define and package the deployment artefacts and processes for the service provider's deployment engineer.

Supporting these capability categories means supporting the following capabilities.

— **Description Development**

1) Contract development – ability to develop contracts that are usable by providers and consumers

2) Business Rules Description Development – ability to develop descriptions of business rules

3) Process Description Development – ability to develop descriptions of processes used to implement services or processes to be deployed by the service provider

4) Description Of Deployment And Execution Environment – ability to describe the deployment environment and execution environment so that the service developed will deploy and operate correctly

— **Implementation Development**

5) Service Provider Implementation Development – ability to implement the service, components or processes needed to implement the service, any operational enablement and deployment enablement

6) Service Consumer Implementation Development – ability to implement consumers of services and ensure that the consumer executes in its hosting environment appropriately

7) Service Hosting Environment Integration Development – ability to develop implementations of services that use the appropriate infrastructure when deployed and executing it the target operating environment

— **Operations Enablement**

8) Provider Monitoring Enablement – ability to develop the implementation of the service so that it can be monitored by the provisioning and service monitoring and management systems in the providers target operating environment according to their policies and governance

9) Provider Management Enablement - ability to develop the implementation of the service so that it can be managed by the deployment, operating, and management systems in the providers target operating environment according to their policies and governance

10) Consumer Management Enablement - ability to develop the implementation of the consuming code such that it can be managed by the consumers monitoring and management systems in the consumers target operating environment according to policies and governance. This also monitors that the provider is meeting his SLA objectives

11) Consumer Monitoring Enablement - ability to develop the implementation of the consuming code such that it can be monitored by the consumers monitoring and management systems in the consumers target operating environment according to policies and governance. This enables the consumer to know if provider SLA's are being met

12) Auditing Enablement – ability to develop the implementation of the service such that it can be audited according to the policies and governance of the target environment. This applies in provider service implementation and consumers client implementations

13) Logging Enablement - ability to develop the implementation of the service such that it can be log events appropriately according to the policies and governance of the target environment to support auditing, tracing, problem determination, billing, SLAs, etc. This applies in provider service implementation and consumers client implementations

14) Governance Enablement - ability to develop the implementation of the service such that it can be governed according to the policies and governance regimen of the target environment. This applies in provider service implementation and consumers client implementations

— **Publication**

15) Discovery – ability to find and register service descriptions and metadata in registries and repositories

16) Subscription To Services Support – ability to develop the implementation of the service that is necessary to support subscription to the service being developed and collection of metrics to enable billing

17) Service Subscription – ability to subscribe to service needed to develop, implement or test the service

18) Event Support for Subscription and Notification of Events – ability to develop services that support subscription to and notification of events

— **Testing**

19) Service Testing – ability to perform functional testing and performance testing of the service

20) Solution Testing – ability to test a set of services that are part of a solution for function and performance

— **Maintenance/Changes**

21) Maintenance – ability to fix errors in a service, package the fixes and send them to the deployment engineer of the service provider

22) Extending Current Implementation – ability to improve or add to the function of the service in order to better meet business requirements. These improvements may be packaged in a deployment package or in a fix package

— **Process Development**

23) Process Implementation – ability to develop process implementations to as part of the implementation of the service

— **Deployment**

24) Deployment Enablement - ability to develop the service implementation and other artefacts so that it can be deployed in the target service provider execution environment

25) Deployment packaging – ability to package the service implementation and other artefacts for deployment in the service providers execution environment

17.1.4 Structural Overview of the Layer

The ABBs in the Development Aspect can be thought of as being logically partitioned into the categories that support:

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

— ability to document descriptions of services and contracts for both provider and consumer,

— ability to develop service implementations for both the provider and consumer,

— ability to develop service implementations that can be managed and governed by the service provider,

— ability to discover and publish services by the provider and consumer,

— ability to test services functionally and for interoperability by the provider and consumer,

— ability to fix, maintain and improve the service by the provider,

— ability to define and develop processes as part of a service implementation by the service provider, and

— ability to define and package the deployment artefacts and processes for the service provider's deployment engineer.

In the diagrams that are used throughout this part of ISO/IEC 18384 to provide a structural overview of the layers of the SOA RA, the ABBs have been colour-coded to match the architecture layers in the to which they belong, and a prefix has been added to the name of the ABB for additional clarity. White indicates ABBs that are defined in this layer. ABBs owned by other layers that are used to support the capabilities of the current layer are shown in darker shades of grey which match the colours of the layers in the SOA RA layers diagram as shown in Figure 3. Each ABB includes one or more numbers in the box which indicate which capabilities in the list in 17.1.3 that the ABB supports. For example, in Figure 63, the ABBs from the Management and Security Aspect are a very dark grey (with a prefix of 'MaS:') while the ABB from the Integration Aspect is shown as black (with a prefix of "Integration"). For example, in Figure 63, the ABBs from the Management and Security Aspect are a very dark grey with a prefix of 'MaS:'. MaS: Access Controller supports capability number 15: '15: Ability to authenticate/authorize for service invocation and message routing'. The Integration: Logger supports capability 9, 10, 11, 13. 13 is '13: Logging Enablement ability to develop the implementation of the service such that it can be log events appropriately according to the policies and governance of the target environment.- to support auditing, tracing, problem determination, billing, SLAs, etc. This is applies in provider service implementation and consumers client implementations'.

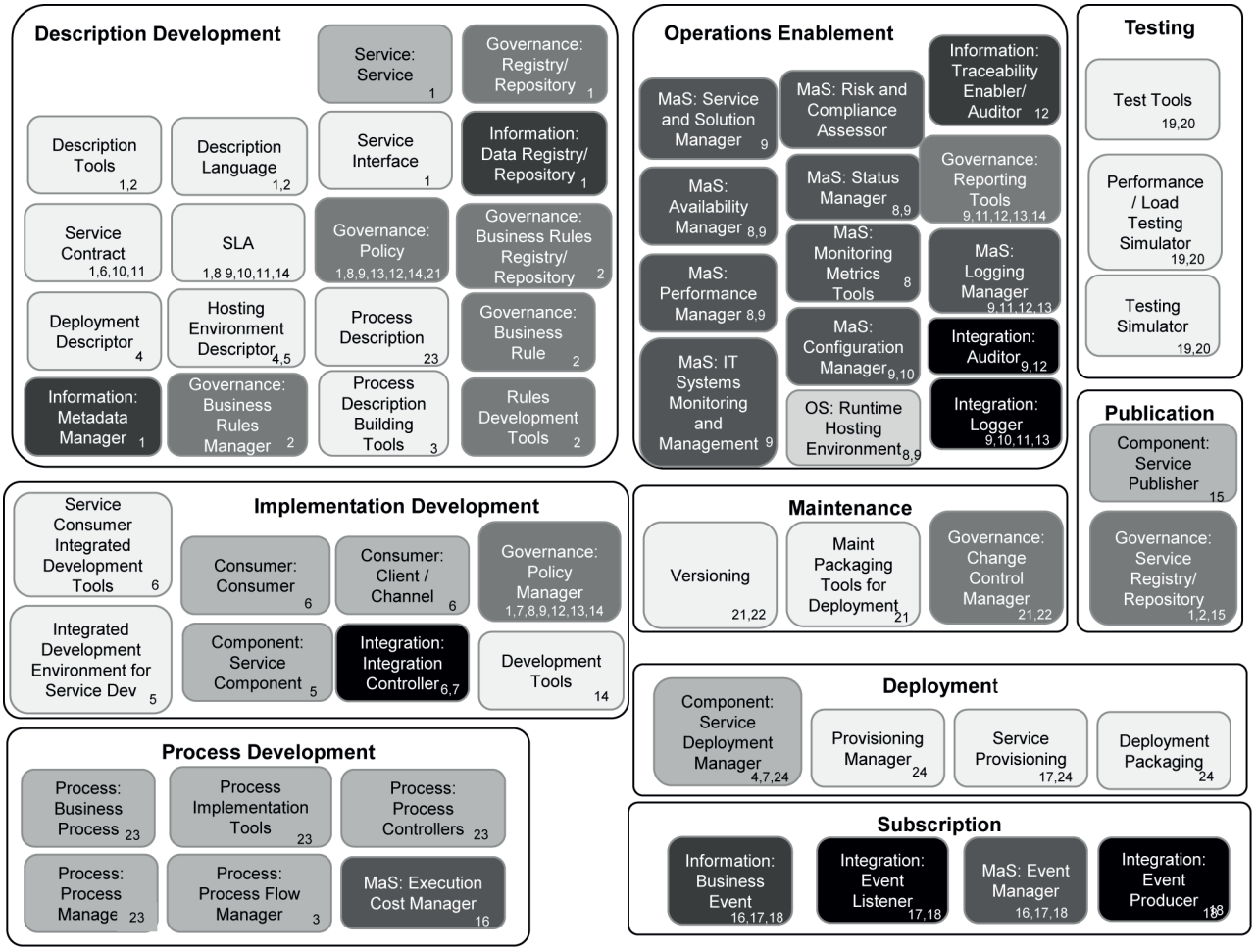


Figure 63 — ABBs in the Development Aspect

The details of the ABBs in 17.2 are grouped by the capabilities.

17.2 Details of ABBs and Supported Capabilities

17.2.1 Description Development

17.2.1.1 Service Layer: Service

See 10.2.1.1.

17.2.1.2 Governance Aspect: Registry/Repository

See 16.2.2.1.

17.2.1.3 Description Tools

This ABB represents the tools used to create any of the description artefacts for services and SOA solutions, including interfaces and policies. Tools may generate descriptions from implementations of the service. It often includes basic Create-Read-Update-Delete operations on the description artefacts.

17.2.1.4 Description language

This ABB represents the language used to capture the description of a service, including the interface to interact with the service, any quality of service requirements for machine processing.

17.2.1.5 Service Interface

This ABB represents the definition of service inputs and outputs and a method of invocation. It is the syntax (how the data is represented) for the contract. It includes event formats for any events the service may include. It can define

— method of invocation,

— error responses or fault conditions, and

— optional element specification patterns (element defined to be nil vs no element defined may have different meanings but is exactly the same on the wire).

17.2.1.6 Information Aspect: Data Registry/Repository

See 15.2.7.1.

17.2.1.7 Service Contract

This ABB represents the specification of the service that a consumer uses to interact with the service. It includes the definition of the service syntax, semantic description of the function of the service and agreements on the quality of service concerns, including security and service levels. It may include other description artefacts: the service interface, the service level agreements, and policies. This specification is agreed to and adhered to by the provider and consumer.

17.2.1.8 Service Level Agreement

This ABB represents the definition of the quality of service of the interactions and performance. It is provided by the Service provider to the Service consumer. It can define

— security constraints,

— reliable messaging and transactional requirements (optional),

— common metrics for availability and performance guarantees, for example, availability time, response time, etc. (optional),

— versions of service (optional), and

— policies that affect the consumer's use of the service.

17.2.1.9 Governance Aspect: Policy

See 16.2.4.1.

17.2.1.10 Governance Aspect: Business Rules Registry/Repository

See 16.2.3.3.

17.2.1.11 Deployment Descriptor

This ABB represents the description of the service components and information needed by the service deployment engineer to take a set of artefacts, including implementations and descriptions and deploy the service implementation into an operational layer or platform and expose the service so it can be used by the consumers. This implies that a mechanism exists so that consumers can find or discover

the service and that mechanism is aware of the service and how the service is designed to be accessed. Optional.

17.2.1.12 Hosting Environment Descriptor

This ABB represents the description of the requirements of the service implementation. The target runtime execution environment and intended operational management of the service may have an impact on the implementation; those impacts and expectations need to be described for the service implementation developer. This may include enablement and instrumentation of metrics and operations for availability and performance monitoring, management, auditing, logging, and configuration. Optional.

17.2.1.13 Process Description

This ABB represents the description of a process in SOA solution.

17.2.1.14 Governance Aspect: Business Rule

See 16.2.3.1.

17.2.1.15 Information Aspect: Metadata Manager

See 14.2.8.2.

17.2.1.16 Governance Aspect: Business Rule Manager

See 16.2.3.2.

17.2.1.17 Process Description Building Tools

This ABB represents a set of tools that supports the development of descriptions of processes and business processes.

17.2.1.18 Rules Development Tools

This ABB represents a set of tools to support the development and documentation of business rules.

17.2.2 Operations Enablement

17.2.2.1 Management and Security aspect: Services and Solution Manager

See 16.2.4.1.

17.2.2.2 Management and security aspect: Availability manager

See 14.2.4.6.

17.2.2.3 Management and Security Aspect: Performance Manager

See 14.2.4.7.

17.2.2.4 Management and Security Aspect: IT Systems Monitoring and Management

See 16.2.3.

17.2.2.5 Management and Security Aspect: Risk and Compliance Assessor

See 14.2.2.6.

17.2.2.6 Management and Security aspect: Status Manager

See 16.2.4.3.

17.2.2.7 Management and Security Aspect: Monitoring Metrics Tools

See 14.2.4.9.

17.2.2.8 Management and Security Aspect: Configuration Manager

See 16.2.8.1.

17.2.2.9 Operational Systems Layer: Runtime Hosting Environment

See 8.2.2.1.

17.2.2.10 Information Aspect: Traceability Enabler/Auditor

See 15.2.3.3.

17.2.2.11 Governance Aspect: Reporting Tools

See 15.2.6.1.

17.2.2.12 Management and Security Aspect: Logging Manager

See 14.2.6.4.

17.2.2.13 Integration Aspect: Auditor

See 13.2.3.4.

17.2.2.14 Integration Aspect: Logger

See 13.2.3.3.

17.2.3 Testing

17.2.3.1 Test Tools

This ABB represents a set of tools necessary to test the functional and operational interfaces of the service.

17.2.3.2 Performance/load Testing Simulator

This ABB represents a set of tools that support the testing of services and solutions for appropriate performance characteristics under a load by simulating the interactions of the system.

17.2.3.3 Testing Simulator

This ABB represents the tools and software used to simulate the solution environment for testing the services and SOA solution before deployment. Implementation Development.

17.2.3.4 Service Consumer Integrated Development Tools

This ABB represents a set of tools used by service consumer implementation developers to implement the interactions with the services according to the contracts and service interface descriptions

17.2.3.5 Consumer Layer: Consumer

See 12.2.1.1.

17.2.3.6 Consumer Layer: Channel

See 12.2.1.2.

17.2.3.7 Governance Aspect: Policy Manager

See 16.2.4.4.

17.2.3.8 Integrated Development Environment (IDE) for Service Development

This ABB represents tools used to define and develop Service Components and associated functional and technical components. This includes service contracts and any other associated service metadata.

17.2.3.9 Component Layer: Service Component

See 9.2.1.1.

17.2.3.10 Integration Aspect: Integration Controller

See 13.2.1.1.

17.2.3.11 Development Tools

This ABB represents the tools used in the plan, define, and implement phases of SOA governance, to document governance policies, and to implement SOA governance metrics, checkpoints, and processes.

17.2.4 Maintenance

17.2.4.1 Versioning

This ABB represents the versioning of services so that services can be updated and maintained and tracked.

17.2.4.2 Maintenance Packaging tools for deployment

This ABB represents a set of tools that support the packaging of code and components for maintenance and fixes into deployment packages that can be deployed onto an existing service and solution in a runtime environment.

17.2.4.3 Governance Aspect: Change Control Manager

See 16.2.6.4.

17.2.5 Publication

17.2.5.1 Component Layer: Service Publisher

See 9.2.2.1.

17.2.5.2 Governance Aspect: Service Registry/Repository

See 16.2.2.3.

17.2.6 Process Development

17.2.6.1 Process Layer: Business Process

See 11.2.1.1.

17.2.6.2 Process Layer: Process Manager

See 11.2.3.2.

17.2.6.3 Process Implementation Tools

This ABB represents a set of tools that supports the development of implementations of processes and business processes.

17.2.6.4 Process Layer: Process Flow Manager

See 11.2.3.4.

17.2.6.5 Process Layer: Process Controllers

See 11.2.3.5.

17.2.6.6 Management and Security Aspect: Execution Cost Manager

See 14.2.4.8.

17.2.7 Deployment

17.2.7.1 Component Layer: Service Deployment Manager

See 9.2.3.1.

17.2.7.2 Provisioning Manager

This ABB represents the management of the installation and instantiation of resources and services into an execution environment.

17.2.7.3 Service Provisioning

This ABB represents the instantiation of resources necessary for the service, followed by the instantiation of the service itself.

17.2.7.4 Deployment Packaging

This ABB represents the assembly of all the artefacts needed to instantiate an executing service in the target runtime environment.

17.2.8 Subscription

17.2.8.1 Information Aspect: Business Event

See 15.2.6.3.

17.2.8.2 Integration Aspect: Event Listener

See 13.2.2.7.

17.2.8.3 Management and Security Aspect: Event Manager

See 14.2.6.1.

17.2.8.4 Integration Aspect: Event producer

See 13.2.2.6.

17.3 Inter-Relationships between the ABBs

In the following diagrams, the arrows between the ABBs indicate an interaction from one ABB to another. A set of ABB relationships for major activities by different roles is explored in this Clause. One set for the Description Developer, for developing the description for the consumer and a description by the service provider for a developer. One set for the service implementer, for developing a consumer and developing a provider.

Figure 64 shows all of the Development Aspect ABBs in the context of their SOA RA functional layers and cross-cutting aspects.

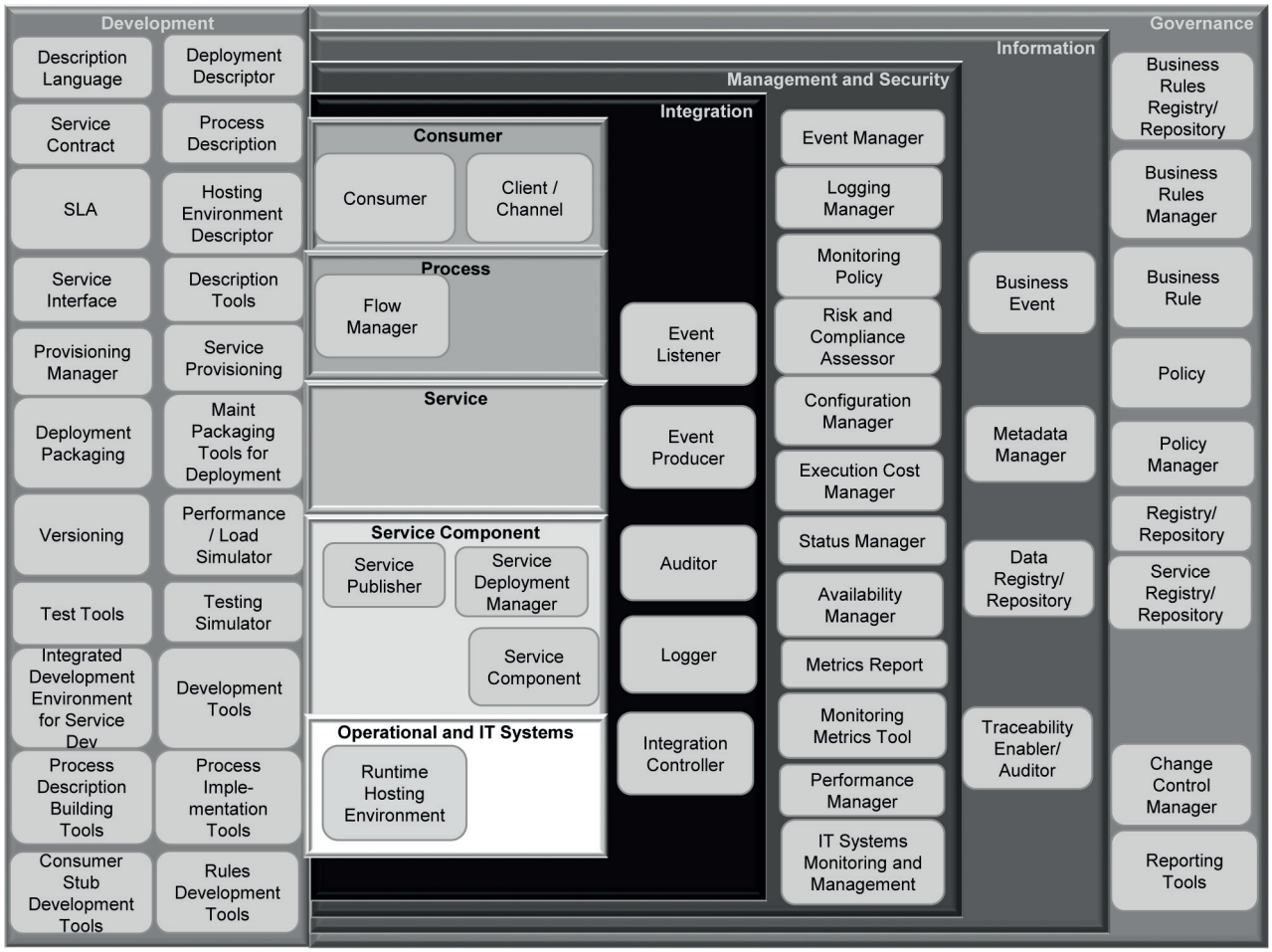


Figure 64 — Relationships among ABBs in the Development Aspect in context of the SOA RA

Figure 65 shows how the interaction between the ABBs supports the development of the description of the service and contract for the Consumer to use in his development. This is typically done by the service provider that will host the service.

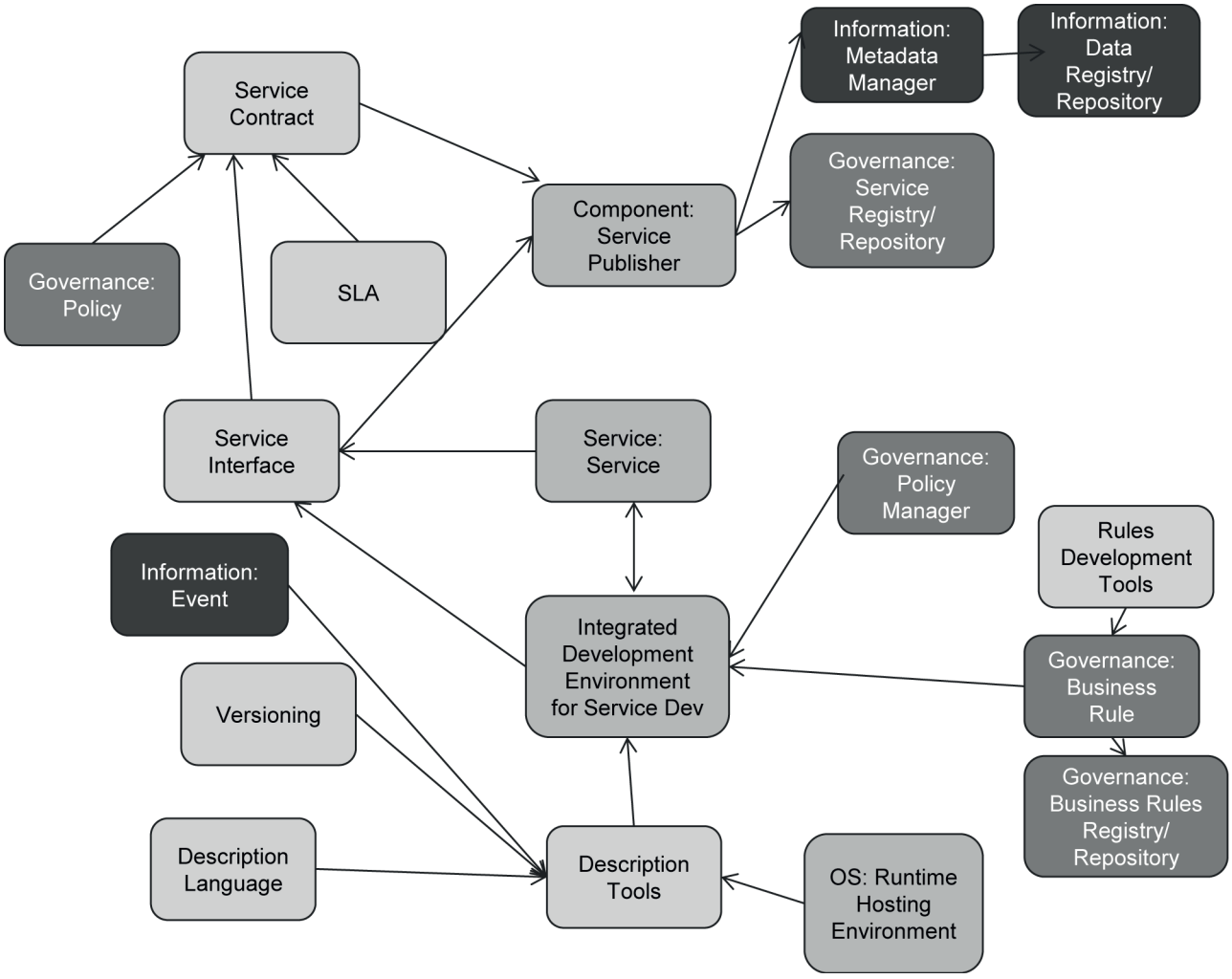


Figure 65 — Creating Service Descriptions for Consumer implementation Developers

Figure 65 shows that a Description language and Description Tools to support that language are chosen. The description of the service interface, events, SLAs, policies and contracts are developed using these tools using information from a variety of inputs. Those inputs to the service description include interfaces generated by the Service Development IDE, policies on the service from the Policy Manager, Business Rules from the Business Rules Repository and Business Rules Manager, and the Runtime Hosting Environment details. A Contract is created for the service Consumer that contains the appropriate Policy, SLA, Events, and Interfaces. The Contract and Service Interface may be published into a service Registry/Repository, as well as a data Registry/Repository.

Figure 66 shows how the interaction between the ABBs supports the development of the description of the service for the developer to use. This creates a set of descriptions needed by the developer of the service and components so that the service will deploy appropriately and execute according the needs of the provider, including operational enablement, logging, auditing, and adhering to governance processes and policies.

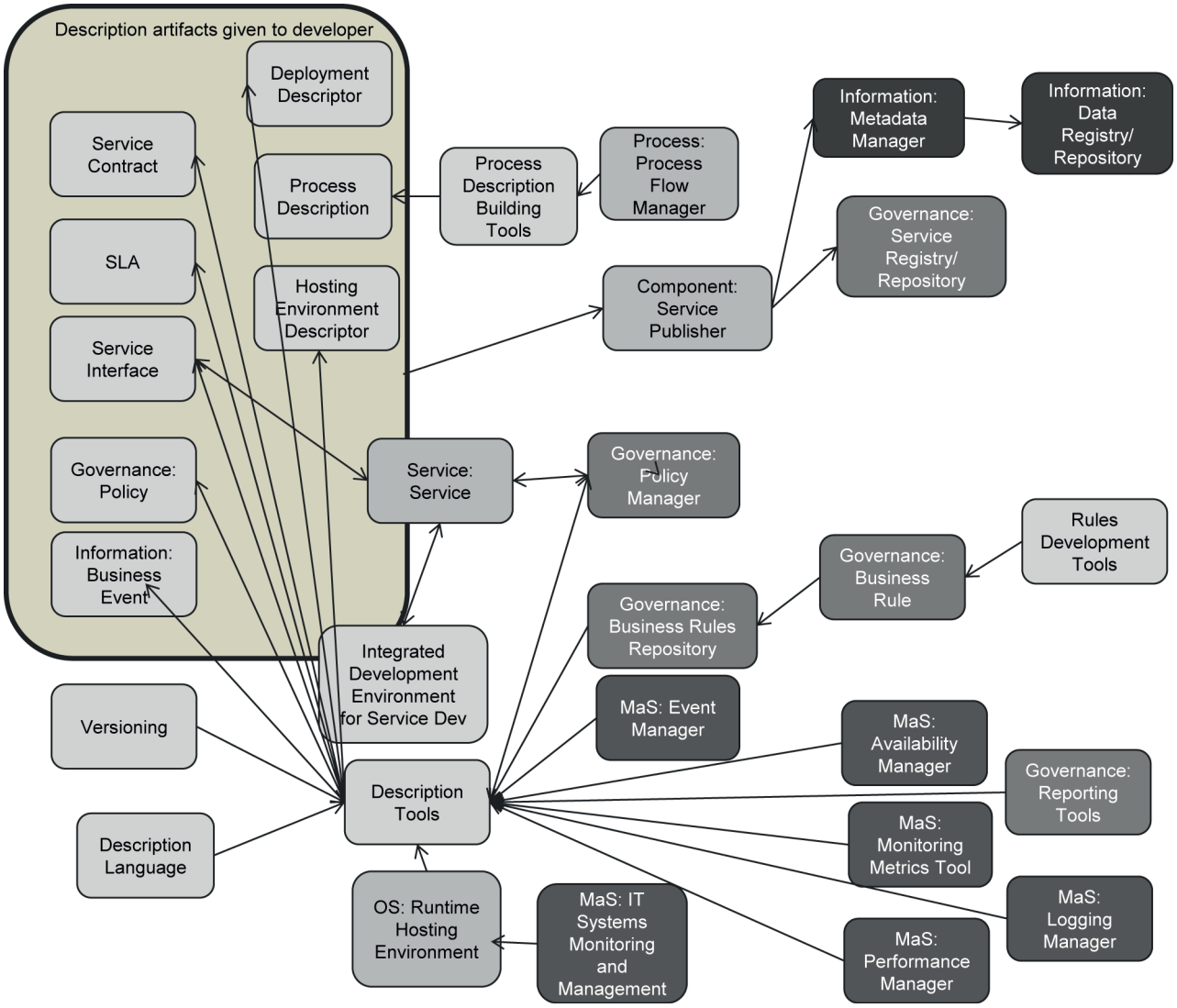


Figure 66 — Creating Service Descriptions for Service implementation Developers

In Figure 66, a Description language and Description Tools to support that language are chosen. Like the description for the Consumer Implementer, the Description of the service interface, events, SLAs, policies and contracts are developed using these tools using information from a variety of inputs. For the service implementation developers, however, those inputs are greatly expanded because the service implementation developer needs much more information to ensure that the service is implemented and packaged correctly. Those inputs to the service description include interfaces generated by the Service Development IDE, policies on the service from the Policy Manager, Business Rules from the Business Rules Repository and Business Rules Manager, and the Runtime Hosting Environment details. In addition, the description includes information about the Runtime Hosting Environment and how it is managed so that the operational enablement is correct. Correct operational policies and descriptors are created for the service implementation developer by consulting the Policy Manager, Business Rules Manager, Availability Manager, Monitoring Metrics Tool, Performance Manager, and Logging Manager. In addition, any Processes that need to be implemented or supported by the service implementation developer are described using Process Description Building Tools and consulting the Process Flow Manager. Once the descriptions are created, the Service Publisher can publish them in the Service Registry/Repository and Data Registry/Repository. In this case, a set of descriptions are provided to the developer and not necessarily assembled into a contract. This set includes contract, SLA, Service Interface, Policy, Events, Process Descriptions and Hosting Environment Descriptors.

The flow in Figure 67 shows how the ABBs support the development of service consumers by consumer implementation developers.

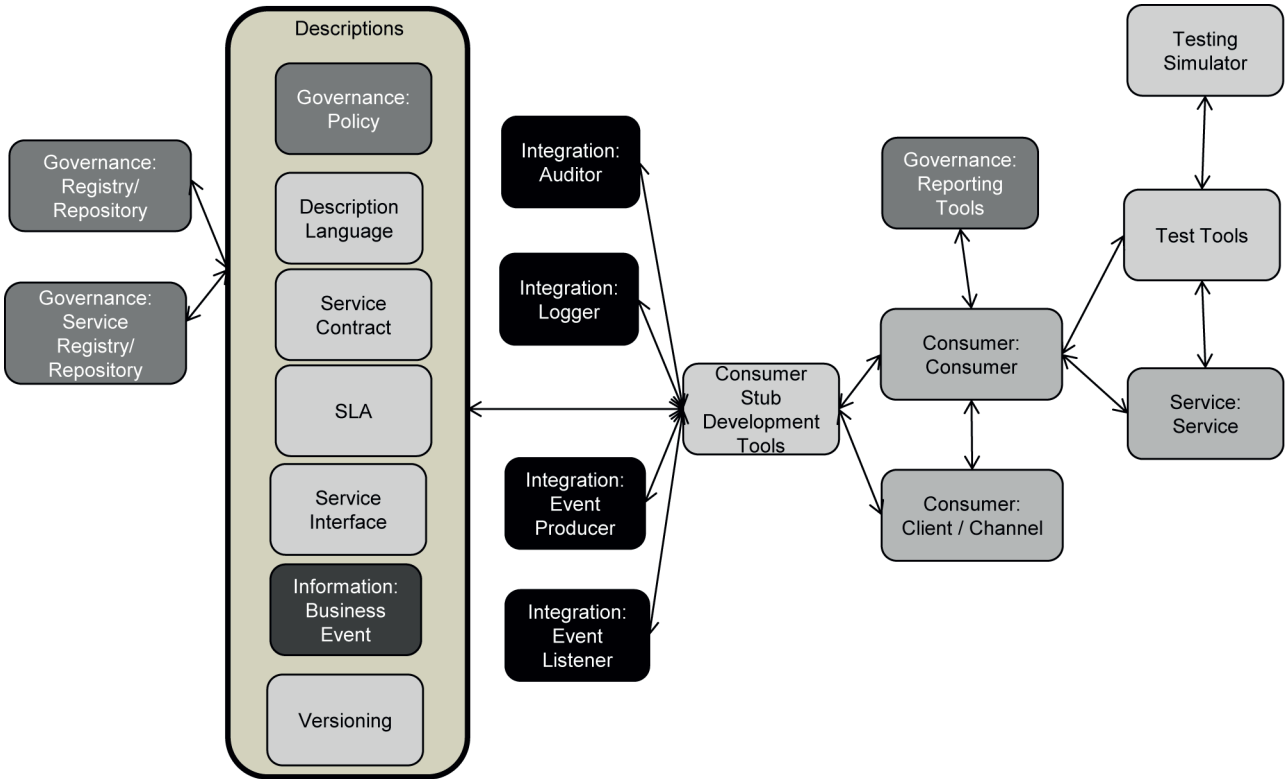


Figure 67 — Development of Service Consumer Implementations

In this flow, the descriptions needed to develop the service consumer are retrieved from a registry/repository, service registry/repository. Alternatively, the descriptions and contract, which may be negotiated unique to each consumer, can be sent to the consumer implementation developer. The consumer implementation developer now uses software development tools and probably tools that will generate the right 'stub' for the consumer to interact with the service. The development of the 'Consumer' as the implementation of the service includes development to support auditing, logging and events as needed by the consumer. An independent Consumer Client or Channel can be developed or generated to isolate the protocols used to interact with the service consumer. The Service Consumer can be tested using Test Tools and a testing simulator, if there is a cost or penalty for testing with the live service. Once the Consumer is functional, it can now interact with the service.

The flow in Figure 68 shows how the ABBs support the development and packaging of services.

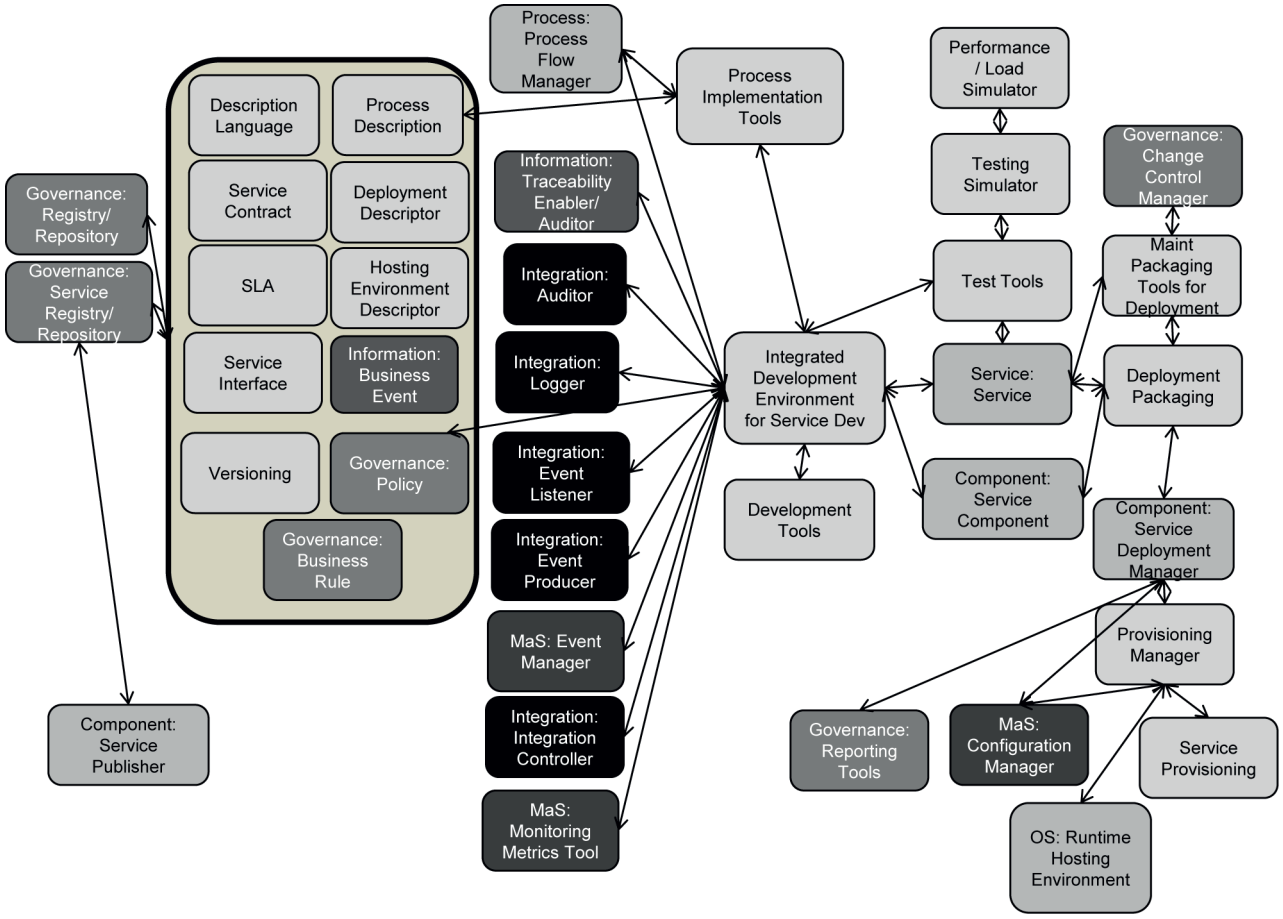


Figure 68 — Development of Service Implementation

In Figure 68, the descriptions can be obtained from the Registry/Repository, Service Registry/Repository or they can be given to the Service Implementation Developer using any other means. Using these descriptions, the Service Implementation Developer will develop the components to implement the service. As part of the service implementation, operational enablement will need to be done. The information needed for enablement should be in the Hosting Environment Descriptor or Policy. According to those requirements, then the service implementation should use the right Logger and Auditor to write the appropriate records to logs. Monitoring Metrics tool may need to be supported in order to enable Performance, Status and Availability Management. In addition, Events support may need to be enabled, either sending or receiving events, therefore, support for the Event Producer and/or Event Listener may be included as part of the implementation. The Integrated Development Environment for Service Development is used to develop the service implementation and all of the needed capabilities of these ABBs. Any development tool can be used to create the implementation. Sometimes, Service implementations need to contain processes as part of the implementation, in that case, the process description is included and a Process Development Tool is used to implement the process using the Process Flow Engine. The Development Tools create Service Components and Service.

The Test Engineer can use the development tools and the Service can use the Test Tools to test the service. The Test Tools can include the Testing Simulator and Performance Load Simulator. Once the Service is implemented and tested then the Service is packaged with all of the artefacts, including code, descriptors, etc. This Deployment Package is given to the Service Provider, who may work for the same company as the Developer or a different company; therefore, it is important that there is a separation of concerns between the development and deployment steps.

During deployment, the Deployment Engineer takes the deployment package and works with the Deployment Manager to arrange for application, infrastructure, and resource installation and

configuration, using the configuration manager onto the Runtime Hosting Environment. Then the Provisioning Manager provisions the services and resources so that the service can be invoked.

When services have an error or need the functionality to be improved then the development tools are used to modify the service components and testing and deployment packaging is performed again. The deployment package for fixes and upgrades may be different than the package for the service.

17.4 Significant Intersection Points with other Layers

17.4.1 Intersection with the Rest of the SOA RA

Figure 69 showed all of the ABBs in the context of the SOA RA layers and Aspects.

17.4.2 Interaction with Cross-Cutting Aspects

The Development Aspect relies on cross-cutting aspects of the architecture to fulfil its responsibilities. These interactions are based on common scenarios and best practices.

It relies on the Governance Aspect for the following capabilities:

— ability to define policy descriptions and access existing policies to support the development of service descriptions and service implementations. Operations enablement is especially dependent on policies for monitoring, management, logging and auditing;

— ability to access and store business rules to support the development of service that adheres to existing business rules when creating service descriptions to reflect those rules;

— ability to store and access service descriptions and metadata in service registries/repositories to support the development of service and service consumer implementations, as well as appropriate description artefacts;

— ability to manage changes in the system since the Development Aspect supports maintenance of services and implementations;

— ability to develop services and solutions that conform to existing governance processes and policies, including reporting and developing the right connections from implementations to governance.

It relies on the Management and Security Aspect for the following capabilities:

— ability to support implementation of operational enablement by consulting with existing application and IT management systems;

— ability to support the implementation of events handling according the event policies of the solution.

It relies on the Integration Aspect for the following capabilities:

— ability to implement support for receiving event subscription requests, producing events an receiving events;

— ability to support implementation of appropriate logging and auditing;

— ability to implement support for the appropriate integration mechanism being used in the Runtime Hosting Environment.

It relies on the Information Aspect for the following capabilities:

— ability to store and retrieve metadata and data as required by service implementation developers and service description developers;

— ability to describe tracing, logging, auditing and event records and formats.

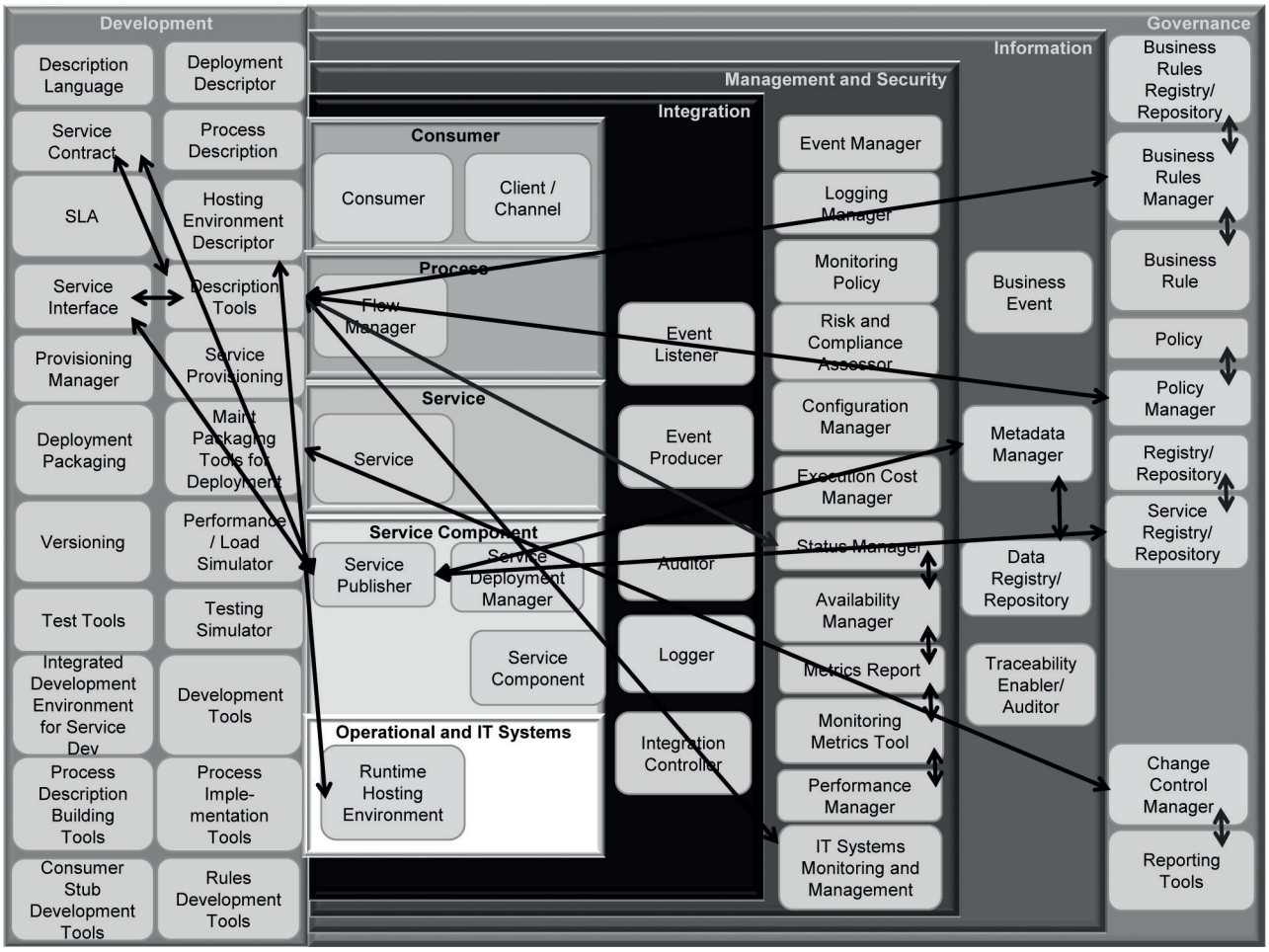


Figure 69 — Key Interactions of Development Aspect with Cross-Cutting Layers

Therefore, Development Aspect interfaces with the following ABBs of cross-cutting layers of the architecture to provide its capabilities.

— It consults the operations management enablers in the Management and Security Aspect when developing the service to ensure it will execute in the runtime hosing environment and adhere to management and governance requirements. These enables that are consulted include Logging Manager, Risk and Compliance Assessor, Status Manager, Execution Cost Manager, Availability Manager, Monitoring Metrics Tool, performance Manager and IT Systems monitoring and management. Monitoring Policy may need to be consulted as well.

— It consults the Event Manager in the Management and Security Aspect so that the correct event APIs and policies are followed for sending and receiving events.

— It defines and uses Event artefacts in the Information Aspect so that the service implementation sends and reacts to the correct events.

— It uses the Metadata manager and Data Registry/Repository in the Information Aspect to store and access the descriptions created to support development and provided by Implementation Developers.

— It needs to enable the implementations for support and audit; therefore, the Traceability Enabler and Auditor ABB is used.

— It relies on the Service Registry/Repository in the Governance Aspect to support for publication of services descriptions and service metadata.

— The Registry/Repository in the Governance Aspect is used to support the publication of artefacts and descriptions developed by Description Developers and Service Implementation Developers.

— It uses the Business Rules themselves, the Business Rules Manager, and the Business Rules Registry/Repository in the Governance Aspect to support access to business rules during the development of descriptions and potentially service operational enablement.

— It relies on the Change Control Manager in the Governance Aspect to support maintenance, testing, and versioning.

— It depends on the Policy and Policy Manager in the Governance Aspect for the policy artefacts needed to create descriptions of services and guide service implementation developers. Policy, in this case, includes that for monitoring, auditing, logging, service level agreements, etc.

The Development Aspect relies on cross-cutting aspects of the architecture to fulfil its responsibilities.

17.4.3 Interaction with Horizontal layers

The Development Aspect provides implementation support for other horizontal layers that are more functional in nature. Each of the other horizontal layers, namely, Consumer Layer, Process Layer, Services Layer, Service Component Layer, have some functional ABBs and some supporting ABBs that are needed to support the implementation of the services. As shown in Figure 70:

— Consumer Stub Development is used to create the Consumer and Client or Channel for the service;

— The Process Description is used by the Process Implementation Tools to create a process which interacts with the Process Flow Manager;

— The Service is tested by the Test Tools, Testing Simulator and/or Performance/Load Simulator;

— The Integrated Development Environment for Service Development is used to create Service Components which called by the service interfaces and deployed by the Deployment Manager. The Deployment Manager creates or uses the Deployment Packaging or Deployment Packing for Maintenance.

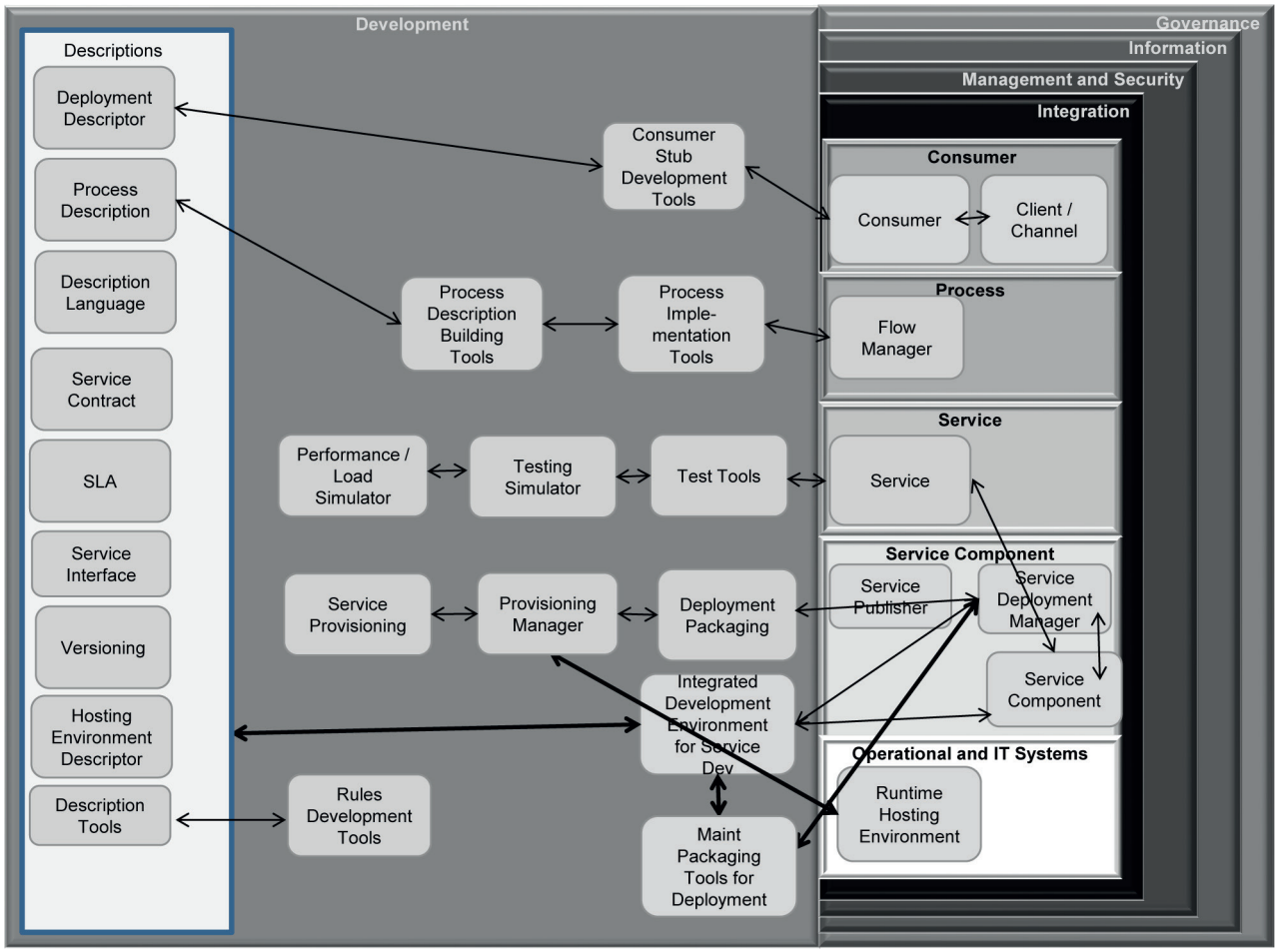


Figure 70 — Key Interactions of Development Aspect with Horizontal layers

17.5 Usage Implications and Guidance

17.5.1 Options and Design Decisions

Table 3 is useful to show the relationships between the roles and the capability categories.

Table 3 — Roles and Capability Mapping for Governance Aspect

Role	Description	Implementation	Process Development	Operational Enablement	Publication	Testing	Deploy	Maintenance
Service Provider	x	x	x	x	x	x	x	x
Service Description Developer	x				x			
Service Provider Implementation Developer		x	x	x	x	x	x	x
Service Consumer Implementation Developer		x			x			
Service Designer; Service Architect	x							
Test Engineer						x		x
Deployment Engineer							x	x
Service Consumer		x						

Table 4 is useful to show the relationships between the roles, capabilities and the ABBs.

Table 4 — ABB to Capability Mapping for Roles for the Development Aspect

#	Viewpoint/Role	Capability	ABB Names	Supported Capabilities Notes
	Capability Category: Description Development			
	Description developer	Contract development	Governance Aspect: Policy Governance Aspect: Policy Manager Governance Aspect: Registry/Repository Governance Aspect: Service Registry/Repository Service: Service Service Interface Information Aspect: Metadata Manager Information Aspect: Data Registry/Repository	Consumer and Provider Implementers agree to and follow this.
	Service provider	Business Rules description development	Rules development Tools Governance Aspect: Business Rule Governance Aspect: Business Rules Registry/Repository Governance Aspect: Business Rules Manager	Business rules should be defined before development of services. However, business rules may affect the descriptions and should be considered by the Description Developer. Optional.
	Provider Implementation Developer	Process description development	Process Descriptions Building Tools (BPMN/BPEL) Process: Process Flow Manager	The Service Provider may need the service implementation to perform some process or be implemented using a process; therefore, the process description would pass TO the Provider Implementation Developer. In addition, the Provider Implementation Developer could choose to implement the service using a process, creating a process description and passing it back from developer to provider. Optional.

Table 4 *(continued)*

#	Viewpoint/Role	Capability	ABB Names	Supported Capabilities Notes
	Provider Implementation Developer Description Developer	Description of deployment and execution environment	Deployment Descriptors Hosting Environment Description Component: Service Deployment Manager	The deployment and execution environment needs to be described and provided to Provider Implementation Developer. Requirements for Operations Enablement are defined here.
	Capability Category: Implementation Development			
	Provider Implementation Developer	Service provider implementation development	Component: Service Implementation Development Environment for Service Development Component: Service Component	Creates stubs for implementations for the Service. Creates integration with deployment and execution environment. Develops implementation to adhere to operational requirements.
	Consumer Implementation Developer	Service consumer implementation development	Service Consumer Integrated Development tools Consumer Aspect: Consumer Consumer Aspect: Channel Integration Aspect: Integration Controller	Creates runtime stubs for the Consumer of the service. Integrate with consumers system (i.e. portal, processes, etc.).
	Provider Implementation Developer	Service hosting environment integration development	Integration Aspect: Integration Controller Component: Service Deployment Manager Governance Aspect: Policy Manager	Ensures the requirements for Operations enablement are described. Ensures that the deployment environment and execution environment is supported by the service implementation. Complies with Service Contract and execution environment descriptions. Understands, leverages, and integrates with the Service Provider's integration ABBs and solution ABBs. Optional.
	Operations Enablement			
	Provider Implementation Developer	Provider monitoring enablement	Service Level Agreement Governance Aspect: Policy Governance Aspect: Policy Manager Management and Security Aspect: Status Manager Management and Security Aspect: Availability Manager Management and Security Aspect: Performance Manager Management and Security Aspect: Metrics Monitoring Tools	Runs on Service Provider. Instruments metrics in the implementation that are required by the Service Provider. Reports metrics according to policy. Instruments to enable status, availability, and performance monitoring. Optional.

Table 4 *(continued)*

#	Viewpoint/Role	Capability	ABB Names	Supported Capabilities Notes
	Provider Implementation Developer	Provider management enablement	Service Level Agreement Governance Aspect: Policy Governance Aspect: Policy Manager Management and Security Aspect: Service and Solution Manager Management and Security Aspect: Status Manager Management and Security Aspect: Availability Manager Management and Security Aspect: Performance Manager Management and Security Aspect: Configuration manager Management and Security Aspect: IT Systems Monitoring and Management Management and Security Aspect: Logging Manager Integration Aspect: Logger Integration Aspect: Auditor Governance Aspect: Reporting Tools	Runs on Service Provider. Enables runtime operations of service: i.e. Start/Stop. Enables configuration of the service implementation. Enables control of runtime/diagnostic log and metrics. Enables /logging and auditing, including log management. Optional.
	Consumer implementation Developer	Consumer management enablement	Service Level Agreement Service Contract. Management and Security Aspect: Configuration manager. Integration Aspect: Logger.	Runs on consumer-side. Enables control of logging, auditing and metrics for the consumer. Enables Configuration, especially for configuration security. Optional.
	Consumer implementation Developer	Consumer monitoring enablement	Service Level Agreement Service Contract Integration Aspect: Logger Management and Security Aspect: Logging Manager Governance Aspect: Reporting Tools	Runs on consumer-side Checks SLAs for compliance. Enables consumer implementation to be monitored. Monitoring services. Optional.
	Provider Implementation Developer	Auditing enablement	Governance Aspect: Policy Governance Aspect: Policy Manager Management and Security Aspect: Logging Manager Governance Aspect: Reporting Tools Integration Aspect: Auditor	Runs on Service Provider. Enables control auditing and recording to Audit Logs for the Service. May require a particular auditing API. Optional.
	Provider Implementation Developer	Logging enablement	Governance Aspect: Policy Governance Aspect: Policy Manager Management and Security Aspect: Logging Manager Governance Aspect: Reporting Tools Integration Aspect: Logger	Run on Service Provider. Enables control of logging records and managing the resulting logs. May require a particular logging API. Optional.
	Provider Implementation Developer Consumer Implementation Developer	Governance enablement	Governance Aspect: Policy Governance Aspect: Policy manager Service Level Agreement Governance Aspect: Reporting Tools Governance Aspect: Development Tools	Runs on Provider-side and Consumer side but they may be completely DIFFERENT governance regimens if they run in different organizations. Ensures service complies with and is part of the Governance Regimen. Enable the Compliance processes for the service. Optional.

Table 4 *(continued)*

#	Viewpoint/Role	Capability	ABB Names	Supported Capabilities Notes
	Publication			
	Developer Description Developer	Discovery	Governance Aspect: Service Registry/Repository Component: Service Publisher	Supports registry of services. Supports finding services (including someone providing the URLS).
	Subscription			
	Provider Implementation Developer	Subscription to services support	Management and Security Aspect: Event manager Management and Security Aspect: Execution Cost Manager Information Aspect: Event	Enables subscription of services being developed so that a consumer can 'subscribe' to the service. Optional.
	Consumer Implementation Developer	Service subscription	Management and Security Aspect: Event Manager Integration Aspect: Event Listener Management and Security Aspect: Service Provisioning Information Aspect: Event	Enables the Consumer implementation to support subscription to services as well as invoke them. This is a common case with cloud services. Optional.
	Provider Implementation Developer	Event support for subscription and notification of events	Integration Aspect: Event manager Management and Security Aspect: Event Manager Integration Aspect: Event Producer Integration Aspect: Event Listener Information Aspect: Event	Enables event driven services to be developed. Enables services to drive events to consumer or other services. Optional.
	Testing			
	Test Engineer	Service testing	Test tools Performance and load testing simulator	Validates the contract. Validates the interactions with the service. Performs functional testing of the service. Performs testing for Performance/and varied loads for the service. Optional.
	Test Engineer	Solution testing	Test tools Performance and load testing simulator	Performs functional testing of the solution. Performs testing for Performance/and varied loads for the solution. Optional.
	Maintenance/Changes			
	Service Provider Deployment Engineer Service Provider Implementation Developer	Maintenance	Governance Aspect: Policy Maintenance Packaging tools for deployment Versioning Governance Aspect: Change Control Manager	Packages maintenance for deployment. Updates version of the service. Optional.
	Service Provider Deployment Engineer Service Provider Implementation Developer	Extending current implementation	Versioning Governance Aspect: Change Control Manager	Updates the service to extend or augment its functionality (essentially is a new service). Optional.

Table 4 *(continued)*

#	Viewpoint/Role	Capability	ABB Names	Supported Capabilities Notes
	Process Development			
	Service Provider Implementation Developer	Process implementation	Process: Process Controllers Process: Process Manager Process: Business Process Process implementation tools	Supports the development of processes that are part of a service implementation. Optional.
	Deployment			
	Service Provider Implementation Developer Deployment Engineer	Deployment enablement	Packaging Provisioning Manager Component: Service Deployment Manager	Supports packaging all of the service implementation and description artefacts for handing to the Service Provider. Packaging is used by a Deployment Engineer to create a Deployment Package or to deploy the implementation directly into the operations. This may also include Provisioning enablement and Retirement enablement. Optional.

18 Common Service Categories

18.1 General

As outlined in Clause 10, one type of ABB in the Service Layer is a service. Services are naturally a key concept in any SOA and it is important to realize that there can be many different kinds. This Clause defines a standard categorization scheme for services. Services are categorized according to what they do, i.e. their function or purpose, in order to aid in ensuring both coverage and shared understanding. Of course, other categorization schemes are also possible and helpful.

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

Note that the categorization scheme for a particular type of ABB is distinct and separate from which layers in the reference architecture such ABBs are associated with. For example, there is no contradiction in defining "process services" as a category of services. A process service is simply process logic exposed as a service. Since these are in fact services, as service ABBs, they belong in the Services Layer. Yet, having said that, their realization as processes properly belongs in the Process Layers and typically leverages other ABBs such as a Process Controller. In fact, the process service implementations will often also rely on implementations of ABBs in other layers, like the Process Manager and policy ABBs in the Management and Security Aspect. The seeming dichotomy is not a dichotomy at all but simply a natural consequence of delineating the service itself (as a service) from the realization of that service (as a process). This is consistent with ISO/IEC 18384-3 where there is a clear delineation between the logical service itself and things that perform that logical service.

Figure 71 shows a functional categorization scheme for services found in a typical enterprise. As mentioned above, this categorization scheme is for the services themselves, not for their implementations (which will be ABBs of other layers of the SOA RA). These categories are loose groups of services that support the same function and where services that support more than one function can exist in more than one category.

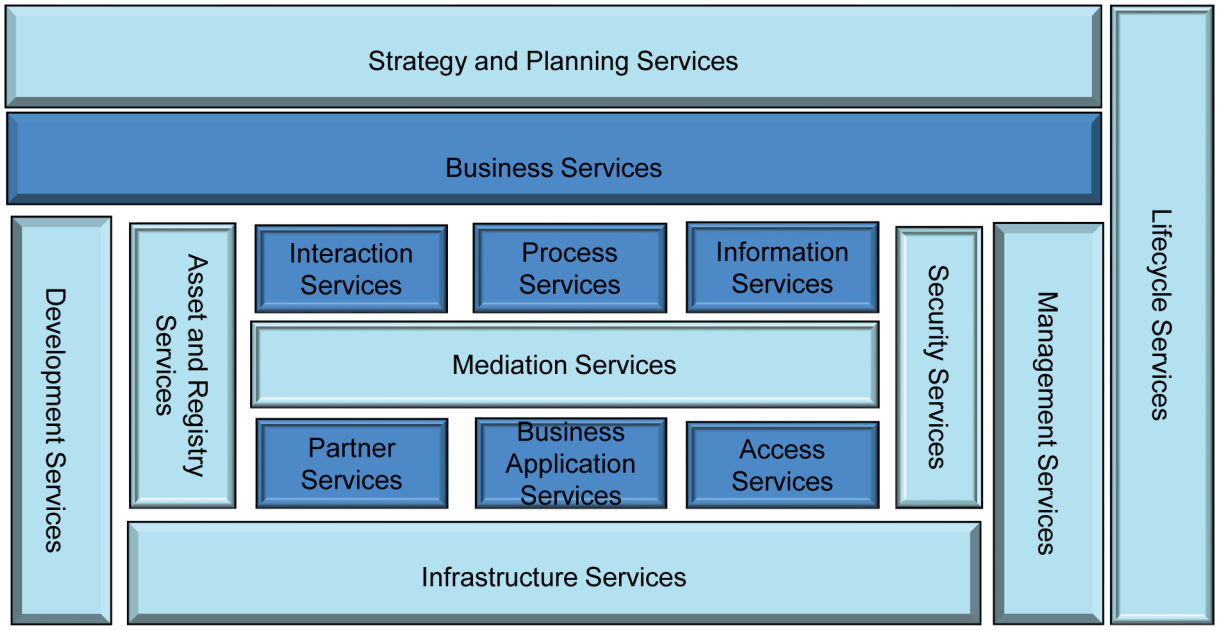


Figure 71 — Functional Categorization Scheme

Categorizations of services are broken down in Figure 71. Dark coloured categories (such as the interaction services, process services, etc.) are considered to be domain-specific. Services in the domain-specific categories are solution-specific and thus require unique implementation-specific ABBs to implement their semantics. Sometimes, domain specific services can be purchased but, generally, they require extensive customization or extension.

The remaining services categories are considered to be domain-neutral. These domain-neutral categories include development services, management services, etc. Services in these categories can be used in any domain or solution. In general, domain-neutral services are used to plan, develop, support, and manage the domain-specific services in the solution. Often, domain-independent services can be purchased and used without extensive customization. However, domain-specific input, such as specific policies, need to be specified as the context on which domain-neutral services are used.

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

Services categories are described in 18.2.

18.2 Mediation Services

Mediation services are a category of service that assumes the responsibility for binding service consumers with service providers; they implement this responsibility by resolving their location automatically to achieve an optimal routing of requests across the network and meet the goals of the business. The presence of mediation services should be transparent to the consumer of functional services in the SOA solution. Though transparent, mediation services are fundamental to simplifying the task of invoking services, making the use of services wherever they are needed. Mediations offered by mediation services leverage the connectivity often do some other activity in addition to the connectivity. Implementations of the mediation services support interconnectivity and host Mediations – logic that may perform message transformation, intelligent routing, augmented functionality (such as logging or auditing) to enable the interconnectivity of services. Because the mediation services are a transparent fabric interconnecting service consumers with service realizations/implementations, then, by extension, the mediation service implementation also makes hosting mediating logic, and more specifically the hosting topology, transparent to the service consumers and providers which are being mediated. Mediation services are a mix of domain-neutral (messaging products and ESBs available from

many vendors) and domain-specific (the implementations of adapters needed from existing services and operational systems into the ESB).

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

18.3 Interaction Services

Interaction services are a category of services that provide the presentation logic of the business design. These services are components that support the interaction between applications and end users. Interactions with the external world are not limited to just interactions with humans; interaction logic orchestrates the interface to all kinds of devices and control systems, including vehicles, sensors, and RFID devices. Every external interaction projects a view of the information system that is tailored for the specific interaction fidelity, frequency of interaction, and presentation composition that best fits the needs of the end user or device.

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

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

18.4 Process Services

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

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

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

Process services are most closely aligned with the process layer. Process service implementations implement or use implementations of the process controller and process flow manager ABBs. In turn, these ABBs in the process layer implement ABBs from other layers, like the business rules manager, where business rules are defined, in the governance aspect.

18.5 Information Services

Information services are a category of services that contain the data logic of business design. The service implementations that provide the data logic have three major functions: to provide access to the persistent data of the business, to support data composition of the business, and to provide their own sub-architecture for managing the flow of data across the organization.

— Data Access: The data access information service implementations can include query statements for retrieving information or referential integrity checks on the information manipulated by these service implementations. Information services for data access incorporate federation of multiple data sources.

— Data Composition: The data composition information service implementations compose information in a way that matches the composition of services in the business design. This is analogous to the kind of re-factoring that can occur with legacy applications to get them to fit better with the business design. In addition, it is common practice to implement these services to separate the database design from the application design to achieve the level of performance and scalability required in many enterprise computing environments.

— Data Flow: The data flow information service implementations manage the movement of information from one part of the enterprise to another. The movement of data is needed to satisfy its own data flow and lifecycle requirements. This may involve the use of Extract-Transform-Load (ETL) mechanisms to process and enrich data in bulk, batch processing activities involved in bulk transaction processing, and migrating data from master-data-of-record databases to information warehouses that can be used to perform post-processing and business intelligence, analytics, and content management functions, which in turn are made available to the business application as services.

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

18.6 Access Services

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

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

18.7 Security Services

Security services are a category of services that address the protection against threats across the vulnerable dimensions of an SOA. The protecting of interactions between service consumers and service providers is only one part of security services. They are also responsible for protecting all of the elements that contribute to the architecture.

Some of the threats that SOA security needs to protect from are the following.

— Destruction (an attack on availability): Destruction of information and/or resources and/or components accessed through services or related to service and service lifecycle.

— Corruption (an attack on integrity): Unauthorized tampering with an asset accessed through services or related to service and service lifecycle.

— Removal (an attack on availability): Theft, removal or loss of information and/or other resources affecting services.

— Disclosure (an attack on confidentiality): Unauthorized access to an asset or a service.

— Interruption (an attack on availability): Interruption of services. Service becomes unavailable or unusable.

Security services for SOA solutions should include those that support

— providing access control and authentication management of related resources,

— providing secure interaction services, including the security protection measures during the transmission process to prevent information tampering, leaking, etc.,

— providing security for the information/data of the services. It should provide encryption and decryption, signature, data integrity verification and other services for information/data and other resources, and

— providing auditing and logging services, including access to audit logs, history and tracking logs, and statistical services.

Security services are most closely aligned with the Management and Security Aspect. Security services implementations implement or use implementations of some of the ABBs in the Management and Security Aspect, including the Command Policy Enforcer and Security Manager. These ABB implementations then rely on the Service Registry/Repository and Policy Manager ABB implementations in the Governance Aspect to help implement the security services

18.8 Partner Services

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

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

18.9 Lifecycle Services

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

Implementations of the lifecycle services rely strongly on asset and Registry/Repository service implementations since these provide access to some of the portfolio of assets that the lifecycle service

implementations manage. The assets that are managed include service implementations, processes, documents, etc.

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

18.10 Asset and Registry/Repository Services

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

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

18.11 Infrastructure Services

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

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

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

Infrastructure services are most closely aligned with the Operational and IT Systems Layer. Infrastructure services implement or use implementations of the Solution Component, Implementation Controller, as well as Hardware and Virtualized Infrastructure ABBs. Infrastructure service implementations implement or use implementations of many of the ABBs in the Management and Security Aspect to provide the management of the infrastructure services and underlying resources, i.e. IT systems manager, availability manager, and performance manager. These ABBs work together to provide an overall IT environment for hosting an SOA solution.

18.12 Management Services

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

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

18.13 Development Services

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

Development services use registry/repository ABBs in the Governance Aspect to get the descriptions needed during development. Development service implementations can also register the appropriate services in the appropriate service registries by using a service in the Asset and Registry/Repository services category or directly via the Service Registry/Repository ABB or one of the specializations, such as the Service Registry/Repository ABB.

18.14 Strategy and Planning Services

Strategy and planning services are a category of services that supports creating a vision, blueprint, and transition plan for improving business outcomes. Specifically, these are services that process the strategies of the business to create an implementation roadmap covering both business and IT. In other words, these services support the long-term evolution and effectiveness of an enterprise.

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

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

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

18.15 Business Application Services

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

Business application services are most closely aligned with the Services Layer. Business application service implementations implement or use implementations of the Service Container and Service Interaction Manager ABBs. Business application service implementations may also implement or use

implementations of ABBs from other layers, including the Access Controller and Policy Enforcer from the Management and Security Aspect, as well as the Policy Manager from the Governance Aspect.

18.16 Business Services

Business services are a category of services that capture the business function and are offered to external consumers. Sometimes, these are referred to as higher level or coarse-grained services.

Business services should be aligned with key performance indicators, business objectives and general metrics, defined and monitored in order to provide business executives, business analysts and other human experts information to ensure the SOA and the service is meeting business objectives.

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

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

18.17 Considering Implementations of Common Service Categories using Reference Architecture

As described in 7.6, the logical representation and descriptions of all services is organized in the services layer, as well as potentially categorized as discussed above. However, some of the category names are common with the Reference Architecture Layer/Aspect names. This is because there is an affinity between the categories and certain layers of Reference Architecture for SOA solutions based on the semantics and where the bulk of the implementation of a service is, as illustrated in the following diagram. The layer association is not exclusive, indeed implementations of these services, just like all other services, use or implement capabilities and building blocks from all of the cross-cutting layers, the Service Component Layer and the Services Layer to fulfil their functional capabilities.

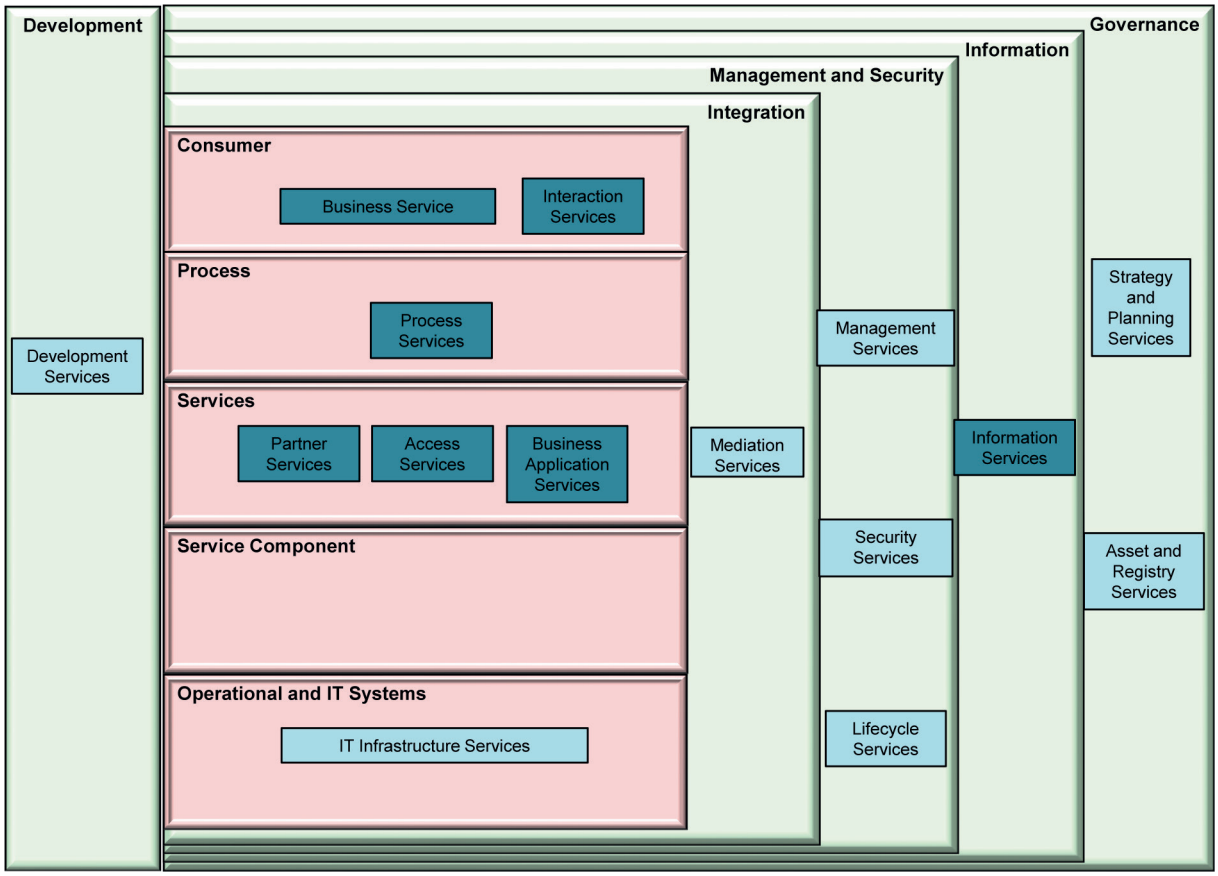


Figure 72 — Affinity between Service Categories and layers of RA for SOA Solutions

For example, in Figure 72, IT infrastructure services provide access to many resources in the Operational and IT Systems layer. Yet, implementations of infrastructure services also use or implement capabilities and ABBs in the Integration Aspect, Management and Security Aspect, Service Component Layer and Services Layer.

Implementations of partner, access, and business application services in the Service Layer provide access to and encapsulation of interactions with third party systems, operational systems and business systems, respectively. These service implementations are domain specific, a.k.a, they offer functionality that is part of the semantics of the SOA solution and are generally implemented specifically for a particular SOA solution.

Implementations of process services implement and provide access to the process layer capabilities, yet are also consumers of partner, access and business application services.

For the Consumer Layer, interaction services support the consumers by supporting interactions through different channels. Business services are services that consumers will interact with directly, yet their implementations are also consumers of other services like process services and business application services.

Mediation services offer many of the capabilities of the Integration Aspect for binding, transforming, and transporting services.

Management services and Security services implementations offer access to the functionality and architectural building blocks in the Management and Security Aspect, like identification management and polling for metrics. Implementations of lifecycle services support tasks like deployment, configuration, change control, enabling services, and disabling services, yet in turn consume asset and registry/repository services.

The Governance Aspect provides business guidelines and policies to be implemented and enforced in the SOA solution. The Strategy and Planning services support (amongst other things) the setting of those guidelines and policies. Asset and registry/repository services are implemented to provide access to registry/repository capabilities in the Governance Aspect.

18.18 Summary

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

19 Related Work and Usages of the SOA RA

The SOA Reference Architecture (SOA RA) provides a mechanism for use in a variety of scenarios:

— for organizations adopting SOA;

— for organizations building SOA components (SOA product vendors);

— for organizations providing services in the construction of SOA (integrators);

— for organizations building standards centred around the specification of SOA standards.

For organizations adopting SOA, it provides a number of uses, including using the SOA RA to create SOA solutions including

— business process-driven,

— tool-based architecture-driven, and

— message-based application integration through

  — service oriented integration,

  — data access-based (information or data services),

  — legacy encapsulation and wrapping, and

  — legacy componentization and transformation.

Applying the SOA modeling and delivery method, every element of SOA that is identified is mapped back to the SOA RA providing a “dashboard view” of the SOA in progress; useful as a communication means for various business and IT stakeholders.

In addition, the SOA RA is used to define capabilities and the technical feasibility of solutions. This usage is focused on a realistic technique of identifying key technical extensible prototypes that test the premises of the architecture and its decisions in a risk-driven fashion.

The SOA RA provides a checklist of key elements that should be considered when building an SOA solution: mandatory, as well as optional layers, attributes, Architecture Building Blocks (ABBs), design decisions, and interaction patterns.

It is important to recognize that SOA solutions are designed and implemented by leveraging existing techniques and technologies. They have an associated set of best practices that are not specifically related to SOA. For example, writing robust Java EE applications and components is an important part of building SOA solutions. This part of ISO/IEC 18384 focuses for the most part on the areas that are critical success factors (CSFs) in building SOAs.

The SOA RA applies to the occupants of various roles within organizations and project teams, including enterprise architects, solution architects, integrators, designers, service modellers and so on. The SOA RA is an abstract, logical design of an SOA. Thus, it answers the question: "What is an SOA?" Architects can use it as a checklist of layers, ABBs, and their relations in each layer, the options available, and decisions that need to be made at each layer. The layers provide a starting point for the separation of concerns needed to build an SOA.

A recurring theme in the context of SOA projects has been the applicability of SOA within multiple areas of increasing scope: a single project, a line of business, a few lines of business-sharing services, enterprise scale, supply-chain (value-net), and a larger SOA ecosystem. In each case, the principles of SOA tend to be applied in a similar manner. This self-similarity of the application of SOA concepts recursively within larger or smaller scopes is termed "fractal" usage of the SOA paradigm.

When SOA is applied, as defined in the SOA RA, to a given level of granularity of an SOA ecosystem, there is typically the need to instantiate ABBs in the same SOA RA layers for each level of granularity. Thus, enterprise architecture might use the SOA RA as an SOA solution template that will be customized or instantiated for each line of business or each product line (depending on how the organization is structured). To participate in an SOA or services ecosystem, an organization would need to have a standard reference architecture such as that depicted by the SOA RA in order to facilitate the integration and collaboration of architectures across organizations. 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.

Bibliography

[1] ISO/IEC 15288, *System life cycle processes*

[2] ISO/IEC 12207, *Systems and software engineering — Software life cycle processes*

[3] ISO/IEC 42010, *Architecture description*

[4] ISO/IEC 10746, *Information technology — Open distributed processing — Reference model: Foundations*

[5] ISO/IEC N0043, *Research Report on China's SOA Standards System*

[6] ISO/IEC N0022, *Chinese National Body Contribution on Proposed NP for General Technical Requirement of Service Oriented Architecture*

[7] OASIS. *Reference Model for SOA*, Version 1.0, OASIS Standard, October 2006: Available from World Wide Web: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[8] The Open Group. *Open Group Standard SOA Reference Architecture Technical Standard,* Available from World Wide Web: http://www.opengroup.org/soa/source-book/soa_refarch/index.htm, pdf format available: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12490

[9] The Open Group. *Technical Standard Service Oriented Architecture Ontology* Available from World Wide Web: http://www.opengroup.org/soa/source-book/ontology/index.htm, pdf format available: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12245

[10] ISO/IEC 17998, *Information technology — SOA Governance Framework*

[11] OMG. *Business Process Management Notation (BPMN),* see http://www.omg.org/spec/BPMN/2.0/

[12] *The Open Group Architecture Framework (TOGAF*), section 8.1.1 Version 9 Enterprise Edition, February 2009; see www.opengroup.org/togaf

[13] OASIS. *Reference Architecture Foundation for Service Oriented Architecture,* Version 1.0, 4, December 2012: see http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf

[14] W3C *Web Services Description Language* (WSDL) 1.1, W3C Note 15 March 2001, see http://www.w3.org/TR/wsdl

[15] OASIS. *Web Services for Remote Portlets Specification* v2.0 OASIS Standard, 1 April 2008 (WSRP), see http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec.html

[16] OMG. *Unified Modeling Language* (OMG UML), Superstructure, Version 2.2, OMG Doc. No.: formal/2009-02-02, Object Management Group (OMG), February 2009: see www.omg.org/spec/UML/2.2/Superstructure

[17] W3C *Web Ontology Language (OWL),* World Wide Web Consortium (W3C), April 2009: see www.w3.org/2007/OWL/wiki/OWL_Working_Group

[18] Garrett J.J. *A New Approach to Web Applications,* Feb 18, 2005 see http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[19] *Web Services Security (WS-Security) Version 1.1 OASIS Standard*, Feb 1, 2006, http://docs.oasis-open.org/wss/v1.1/

[20] *OASIS Web Services Coordination (WS-Coordination) Version 1.2, OASIS Standard,* Feb 2, 2009, http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf

[21] *Web Services Atomic Transaction (WS-Atomic Transaction) Versions 1.2 OASIS Standard*, Feb 2, 2009, http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os.pdf

[22] *Web Services Business Activity (WS-Business Activity) Version 1.2 OASIS Standard,* Feb 2, 2009, http://docs.oasis-open.org/ws-tx/wstx-wsba-1.2-spec-os.pdf

[23] *Microsoft. net*, July 29, 2014, http://www.microsoft.com/net

[24] *Java Platform Enterprise Edition (Java EE),* July 29, 2014 http://www.oracle.com/technetwork/java/javaee/overview/index.html

[25] *XSL Transformations (XSLT) Version 2.0*, January 23, 2007, http://www.w3.org/TR/xslt20/

[26] *Voice Extensible Markup Language (Voice XML) Version 2.0*, March 16, 2004, http://www.w3.org/TR/voicexml20/

[27] *IT Infrastructure Library (ITIL),* July 29, 2014, http://www.itil.org/en/vomkennen/itil/index.php

[28] *Reliability, Availability and* Serviceability (RAS), July 29, 2014, http://dictionary.reference.com/browse/reliability,+availability,+serviceability

[29] ISO/IEC 17789, *Information technology — Cloud computing — Reference architecture*

[30] ISO/IEC/TR 30102, *Information technology — Distributed Application Platforms and Services (DAPS) — General technical principles of Service Oriented Architecture*

ICS 35.100.05

Price based on 191 pages