Document Type: Working Draft

Document Title: ISO/IEC WD **18384**, Distributed Application Platforms and Services (DAPS) - Reference Architecture for Service Oriented Architecture (SOA RA) Part 1

Source: SC38

Document Status: **Accompanying Ballot for Comments on WD 18384 in SC38.**

Action ID: Ballot

Secretariat, ISO/IEC JTC 1/SC 38, American National Standards Institute, 25 West 43rd Street, New York, NY 10036; Telephone: 1 212 642 4904; Facsimile: 1 212 840 2298; Email: mpeacock@ansi.org

Reference number of working document: **ISO/IEC JTC 1/SC 38 N 780**

Date: 2011-10-21

Reference number of document: **ISO/IEC WD 18384**

Committee identification: ISO/IEC JTC 1/SC 38/WG 2

Secretariat: ANSI

Distributed Application Platforms and Services (DAPS) – Reference Architecture for Service Oriented Architecture (SOA RA)

Document type: Technical Report
Document subtype: if applicable
Document stage: (40) Enquiry
Document language: E

Copyright notice

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this Working Draft may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC WD 18384-1 was prepared by Technical Committee ISO/JTC 1, Subcommittee SC 38, *SC DAPS* Work Group 2, SOA Working Group.

ISO/IEC WD 18384 consists of three parts, under the general title: Reference Architecture for Service Oriented Architecture

Introduction

Service Oriented Architecture (abbreviated SOA) is an architectural style that supports service orientation and is a paradigm for business and IT (Note: see 3.1.40). 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 repeatable business activity that has specified outcomes, is self contained, may be composed of other services and is a "black box" to consumers of the service. (Note: see 3.1.14).

To enable this co-operation and collaboration 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 ever-changing business needs. Many companies across many industries world-wide have developed SOA enterprise architectures, solutions and products.

This document 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, improve the standardization and quality of solutions, as well as promote effective large-scale adoption of SOA. The benefits of this technical report contribute to improving the standardization, interoperability, and quality of solutions supporting SOA

This document defines the basic technical principles and reference architecture for SOA rather than being focused on the business aspects. SOA does enable interactions between businesses without specifying aspects of any particular business domain. It also discusses the functional, performance, development, deployment, and governance aspects of SOA. This technical report 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.

This document includes the following clauses:

Clause 3 – terminology – defines terms used when discussing or designing service oriented solutions. Terms defined here are used in some unique fashion for SOA. It does not define terms that are used in general English manner.

Clause 4 – Concepts and Principles – articulates basic SOA concepts and expands on the key terms in clause 3.

The targeted audience of this technical report includes, but is not limited to, standards organizations, architects, SOA service providers, SOA solution and service developers, and SOA service consumers who are interested in adopting and developing SOA.

Distributed Application Platforms and Services (DAPS)

Reference Architecture for Service Oriented Architecture

1 Scope

This working draft describes the general technical principles underlying Service Oriented Architecture (SOA), including principles relating to functional design, performance, development, deployment and management. It provides a vocabulary containing definitions of terms relevant to SOA.

It includes a domain-independent technical framework, addressing functional requirements and non-functional requirements.

2 Terms, Definitions, Notations, and Conventions

For the purposes of this technical report, the following terms and definitions apply

2.1 Definitions

2.1.1

actor

A person or system component who interacts with the system as a whole and who provides stimulus which invoke actions. (Note: see ISO/IEC 16500-8:1999, 3.1])

2.1.2

architecture

Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution ISO/IEC/IEEE 42010:2011, 3.2).ISO/IEC 40210:2011

2.1.3

choreography

Composition whose elements interact in a non-directed fashion with each autonomous member knowing and following an observable predefined pattern of behavior for the entire (global) composition. (Note: see Bibliography Reference [21])

collaboration

Composition whose elements interact in a non-directed fashion, each according to their own plans and purposes without a predefined pattern of behavior. NOTE: See Bibliography Reference [21]

2.1.4

composition

Result of assembling a collection of things for a particular purpose. NOTE: See Bibliography Reference [21]

2.1.5

effect

Outcome of an interaction with a service

Note: If service contracts exist, they usually define effects. The effect is how a service, through the element that performs it, delivers value to its consumer. NOTE: See Bibliography Reference [21]

2.1.6

element

Unit that is indivisible at a given level of abstraction and has a clearly defined boundary,

Note: An Element can be any type of entity NOTE: See Bibliography Reference [21]

2.1.7

entity

Individual in a service system with an identity which can act as a service provider or consumer.

Note: Examples of entities are organizations, enterprises and individuals, software and hardware.

2.1.8

event

Something that occurs to which an element may choose to respond.

Note: Events can be responded to by any element and events may be generated (emitted) by any element.

2.1.9

execution context:

Set of technical and business elements that form a path between those with needs and those with capabilities and that permit service providers and consumers to interact.

Note: The execution context of a service interaction is the set of infrastructure elements, process entities, policy assertions and agreements that are identified as part of an instantiated service interaction, and thus forms a path between those with needs and those with capabilities. NOTE: See Bibliography Reference [19]

2.1.10

human actor

Person or an organizational entity.

Note: In principle, this classification is not exhaustive. NOTE: See Bibliography Reference [21]

2.1.11

human tasks

Tasks which are done by people or organizations, specifically instances of Human Actor.

2.1.12

information Type

Type of information given or received in a service interface.

2.1.13

orchestration

Composition for which there is one particular element used by the composition that oversees and directs the other elements.

Note: the element that directs an orchestration by definition is different than the orchestration (Composition instance) itself. NOTE: See Bibliography Reference [21]

2.1.14

process

Composition whose elements are composed into a sequence or flow of activities and interactions with the objective of carrying out certain work.

Note: A process may also be a collaboration, choreography, or orchestration. NOTE: See Bibliography Reference [21]

2.1.15

REST

Architectural style for distributed hypermedia systems. REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems.

(Note: See REST "Fielding, Roy Thomas (2000), Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine)

2.1.16

service

Logical representation of a set of repeatable activities that has specified outcomes, is self-contained, may be composed of other services, and is a “black box” to consumers of the service NOTE: See Bibliography Reference [21]

Note: The word “activity” in the ‘Service’ definition above is used in the general English language sense of the word, not in the process-specific sense of that same word (i.e., activities are not necessarily process activities).

2.1.17

service broker

Implements service intermediaries that provide unified service registration and publishing.

Note: They can also provide other important supports for SOA, such as service discovery, routing, location-transparent service access, for service providers and service consumers.

2.1.18

service bus

Intermediary IT infrastructure that supports service access and consumption, event-driven message routing among services.

Note: The core functionalities of Service Bus might include: service routing, message transformation, event handling, providing service call, and related intermediary services, connecting a variety of applications, services, information, and platform resources. Service bus is widely used in enterprise contexts and usually equates to the Enterprise Service Bus (ESB).

2.1.19

service catalogue

service registry

service repository

Component that supports publication, registration, search, and retrieval of metadata and artifacts for services.

Note: A service registry is typically a limited set of metadata to facilitate interaction with services and accessing content from a service repository containing the full artifacts.

2.1.20

service choreography

Composition whose elements are services that interact in a non-directed fashion with each autonomous member knowing and following an observable predefined pattern of behavior for the entire (global) composition. NOTE: See Bibliography Reference [21]

2.1.21

service collaboration

Composition whose elements are services that interact in a non-directed fashion, each according to their own plans and purposes without a predefined pattern of behavior. NOTE: See Bibliography Reference [21]

2.1.22

service component

Element that implements services

2.1.23

service composition

service assembly

Result of assembling a collection of services to achieve a particular purpose

Note: A composition can support different composition patterns: such as. collaboration, choreography, orchestration NOTE: See Bibliography Reference [21]

2.1.24

service consumer

Entity that uses services.

Note: Consumers may interact with services operationally or with contractually (legal responsibility).

2.1.25

service contract

Terms, conditions, and interaction rules that interacting consumers and providers must agree to (directly or indirectly).

Note: A service contract is binding on all participants in the interaction, including the service itself and the element that provides it for the particular interaction in question. NOTE: See Bibliography Reference [21]

2.1.26

service description

Information needed in order to use, or consider using, a service.

Note: The service description usually includes the service interfaces, contracts, and policies. NOTE: See Bibliography Reference [19]

2.1.27

service deployment

Process that makes implementations of services able to run in a specific hardware and software environment.

2.1.28

service development

service implementation

Technical development and physical implementation of the service that is part of a service lifecycle.

2.1.29

service discovery

Process that service consumers use to search and retrieve desired services according to their specific functional or non-functional requirements.

2.1.30

service governance

Strategy and control mechanism definition on service lifecycle, which includes establishment of chains of responsibility, ensures its compliance with policies by providing appropriate processes and measurements.

Note: Aspects of the service lifecycle that needs to be governed includes: addressing service modifications, version updates, notice of termination, decomposition subdivision, agency capacity, decomposition capacity, ability to meet individual demands

2.1.31

service interaction

Activity involved in making using of a capability offered, usually across an ownership boundary, in order to achieve a particular desired real-world effect. NOTE: See Bibliography Reference [19]

2.1.32

service interface

means by which other elements can interact and exchange information where form of the request and the outcome of the request is in the definition of the service NOTE: See Bibliography Reference [21]

2.1.33

service interoperability

Ability of providers and consumers to communicate, invoke services and exchange information at both the syntactic and semantic level.

2.1.34

Service Level Agreement (SLA)

Service contract that defines the interaction and measureable conditions of interaction between a service provider and a service consumer.

Note: A Service Level Agreement usually contains: the set of services the provider will deliver, a complete, specific definition of each service, the responsibilities of the provider and the consumer, the set of metrics to determine whether the provider is delivering the service as promised, an auditing mechanism to monitor the service, the remedies available to the consumer and provider if the terms of the SLA are not met, and how the SLA will change over time

2.1.35

service lifecycle

Set of phases for a service throughout its life, from identification to instantiation and retirement.

2.1.36

service management

Monitoring, controlling, maintaining, optimizing, and operating services

2.1.37

service modeling

Service oriented analysis process of identifying and modelling a series of service candidates for functions or actions which can be defined independently or by decomposing business processes.

2.1.38

service monitor

Monitoring and controlling operational state and performance of service.

2.1.39

service orchestration

Composition of services for which there is one particular element of the composition that oversees and directs the other elements.

Note: the element that directs an orchestration by definition is different than the orchestration (Composition instance) itself. NOTE: See Bibliography Reference [21]

2.1.40

service orientation

Approach to designing systems in terms of services and service-based development.

2.1.41

service oriented analysis

Preparatory information gathering steps that are completed in support of a service modeling sub-process that results in the creation of a set of service candidates.

Note: Service Oriented Analysis is the first phase in the cycle, though the service-oriented analysis process might be carried out iteratively, once for each business process. It provides guidance to the subsequent phases of the SOA lifecycle.

.

2.1.42

service oriented architecture

Architectural style that supports service orientation and is a paradigm for building business solutions using IT.

Note: Services realized in this style utilize activities that comprise business processes, have descriptions to provide context may be implemented via service composition, have environment-specific implementations which are described in the context that constrains or enables them, require strong governance, and place requirements on the infrastructure to achieve interoperability and location transparency using standards to the greatest extent possible. NOTE: See Bibliography Reference [21]

2.1.43

SOA governance

Extension of IT governance specifically focused on management strategies and mechanisms for the end users' specific SOA solution.

Note1: It manages the entire SOA lifecycle by setting out personnel, roles, management procedures and decision-making. SOA governance needs to adopt the appropriate methodology and best practices. SOA governance usually requires tools for assistance to customize and manage the governance strategy according to the needs.

Note2: While management means the specific process for governance and control to execute the policies, governance looks at assigning the rights to make decisions, and deciding what measures to use and what policies to follow to make those decisions.

2.1.44

SOA implementation

Process methods and techniques used to develop SOA based solutions.

2.1.45

SOA lifecycle

Process for engineering SOA-based solutions, including analysis, design, implementation, deployment, test and management.

2.1.46

SOA management

Measurement, monitoring, and configuration of the entire lifecycle of a SOA solution.

Note: At runtime it is the process for the specific measurement and operation of the implementation of the SOA solution according to the strategies and mechanisms identified by the SOA governance process.

2.1.47

SOA maturity

Quantitative description an organization’s ability to adopt SOA and the level of SOA application adoption within an IT architecture in an organization.

2.1.48

SOA maturity model

Framework and method to evaluate an organisations’ SOA maturity against overall objectives.

2.1.49

service policy

Statement that an entity may intend to follow or may intend that another entity should follow. NOTE: See Bibliography Reference [21]

2.1.50

service provider

Entity providing services [2.1.16].

Note: Providers may be responsible for the operation of the services or the contract for the service (legal responsibility).

2.1.51

service publishing

Publishing information for registered services making services visible and available to potential consumers.

2.1.52

SOA resource

Elements that provide the IT resources used by services.

2.1.53

SOA solution

Solutions implemented by applying SOA concepts, methods, and techniques.

2.1.54

SOAP

Stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information (Note: See SOAP - SOAP Version 1.2 Part 0: Primer http://www.w3.org/TR/soap12-part0/)

2.1.55

task

Atomic action which accomplishes a defined result. NOTE: See Bibliography Reference [21]

2.1.56

Web Services

Software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. (Note: see Web Services Architecture http://www.w3.org/TR/ws-arch/)

2.2 Acronyms

ABB - Architectural Building Block

BMM – Business Motivation Model (see OMG)

BPMN – Business Process Management Notation

IT – Information Technology

EA – Enterprise Architecture

RA – Reference Architecture

SLA – Service Level Agreement

SOA - Service Oriented Architecture

SOSE – Service Oriented Software Engineering

SQL – Structured Query Language

WSDL – Web Services Description Language

WSRP – Web Services Remote Portlet

KPI – Key Performance Indicator

2.3 Notations

2.4 Conventions

3 SOA Principles and Concepts

3.1 Introduction to SOA

Service Oriented Architecture (abbreviated SOA) is an architectural style that supports service orientation and is a paradigm for business and IT (Note: see 3.1.40). 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 repeatable business activity that has specified outcomes, is self-contained, may be composed of other services and is a “black box” to consumers of the service. (Note: see 3.1.14).

As a foundation for understanding, SOA is an architectural style that has the following distinguishing characteristics:

1. It is based on the design of the services and processes – which mirror real-world business activities – comprising the enterprise (or inter-enterprise) business processes.

2. Service representation utilizes business descriptions to provide context (i.e., business process, goal, rule, policy, service interface, and service component) and implementations of services are provided use processes and service composition.

3. It places unique requirements on the infrastructure – it is recommended that implementations use open standards to realize interoperability and location transparency.

4. Implementations are environment-specific – they are constrained or enabled by context and must be described within that context.

5. It requires strong governance of service representation and implementation.

6. It requires a criteria to determine what a “good service”. (Note: see [20])

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.

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 ever-changing business needs.

In recent years, SOA has become a business organization and technology hot spot that is recognized and respected. Many companies have developed SOA enterprise architecture, solutions and products world-wide. At the same time, an increasing number of solutions are being implemented using SOA in many different industries.

However, 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, which has added to the confusion about the effectiveness of SOA. To improve standardization and quality of solutions, as well as increase effective large-scale adoption of SOA, it is necessary to establish a unified set of general technical principles for SOA.

It should be noted that these SOA principles defined here 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).

The engineering of SOA based systems and solutions, 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).

3.2 Concepts

3.2.1 Roles

Providers

A service provider is an entity providing services. Providers can be responsible for providing services in two different ways:

- Operationally - the provider is responsible for responding to the exchange of messages with the consumer as well as producing the promised effect of invoking the service. Assuming the operational responsibility for providing a service implies the following across the lifecycle of said service:
  - Service Creation: Creating a service implementation that can provide the service in question
  - Providing Services: Providing the implemented service for use by others
  - Publish Service: Publishing service descriptions (its interface and access information) to the service registry
  - Hosting Services: providing a service container to support the runtime service interactions
  - Governance: setting up business rules and policies for lifecycle management of the service

- Contractually - the provider is the entity that participates in the service contracts with the service consumer and is legally obligated to providing the service. Assuming the contractual obligation for providing a service implies the following across the lifecycle of said service:
  - Categorize Service: Deciding what category the service should be listed in for a given service registry
  - Service Pricing: Deciding how to price the services, or how/whether to exploit them for other value
  - Publish Service: Publishing the availability of the service as well as its promised effect
  - Defining Service Agreements: Decide what sort of formal agreements are required to use the service
  - Defining Service Contracts: Setting and abiding by the contracts.
  - Governance: setting up business rules and policies for the service offerings

Sometimes, the entity that is contractually responsible (a participant in a contract or service level agreement) is not the SAME entity that is operationally responsible (i.e. exchanges messages with the consumer)

Note: Consume and Provide: One of the challenges with the service providers and service consumers terminology is that often consuming and providing service is a role in a particular interaction or contractual context. It also does not distinguish between the contractual obligation aspect of consume/provide and the interaction aspect of consume/provide. A contractual obligation does not necessarily translate to an interaction dependency, since it may have been sourced to a third party. It may be more appropriate to work in terms of operationally or contractually consume and provide rather than consumer or provider

Consumers

A service consumer is an **e**ntity that uses services. Consumers will use services in two ways:

- Operationally - the consumer is responsible for the discovery and initiation of exchange of messages with the service. Some of the responsibilities include:

   - Service Discovery: Searching for the most suitable service by examining available service descriptions
   - Service Registry Search: Searching for appropriate services in a given service registry to find the candidate services
   - Service Invocation: Invoke a service by sending it a message

- Contractually - the consumer is the entity that participates in the service contracts with the service provider. Some of the responsibilities include:
   - Contracts: setting and abiding by the contracts.
   - Payments: Paying for services
   - SLAs: Ensuring that services adhere to the service level agreements
   - Governance: ensuring business requirements are met by the usage of the service

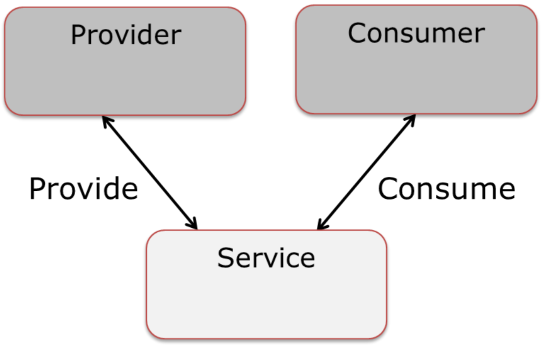

Figure 1: Provider and Consumer Roles

Figure 1 shows that Provider and Consumer are roles that provide and consume services.

3.2.2 Services

As defined in this technical report, a service is a logical representation of a set of repeatable activities that has specified outcomes, is self-contained, may be composed of other services, and is a “black box” to consumers of the service (Note: See [21]). *Service* is agnostic to whether the concept is applied to the business domain or the IT domain. A service can have one or more providers or consumers, and produces outcomes that are of value to its consumers.

To a consumer, a service is a black box, in other words, the consumer does not know how the service is implemented. If two services have the same service contract and when given the same inputs will produce the same effects, they are equivalent to the consumer and should be able to be used interchangeably. To a provider, a service is a means of exposing capabilities and the implementation determines equivalency. Therefore, two services that have the same inputs and produce the same effects but use different mechanisms are not equivalent. .

As a service itself is only a logical representation, any service is *performed* by something. The something that *performs* a service must be opaque to anyone interacting with it. Services can be performed by elements of other types than systems. This includes elements such as software components, human actors, and tasks.

Likewise, a service can be *used by* other elements, the service itself (as a purely logical representation) does not *use* other elements. However, the thing that performs the service might very well include the use of other elements (and certainly will in the case of *service composition*).

An element using a service by interacting with it will perform the following typical steps:

- Pick the service to interact with (this statement is agnostic as to whether this is done dynamically at runtime or statically at design and/or construct time)

- Pick an element that performs that service (in a typical SOA environment, this is most often done “inside” an Enterprise Service Bus (ESB))

- Interact with the chosen element (that performs the chosen) service (often also facilitated by an ESB)

Concepts, such as service mediations, service proxies, ESBs, etc. are natural to those practitioners that describe and implement the operational aspects of SOA systems. All of these can be captured as an element representing the service – a level of indirection that is critical when we do not want to bind operationally to a particular service endpoint, rather we want to preserve loose-coupling and the ability to switch embodiments as needed. Understanding what a service *represents* and is *representedBy* allows encapsulation of the relatively complex operational interaction pattern that was described in the clause above (picking the service, picking an element that performs the service, and interacting with that chosen element).

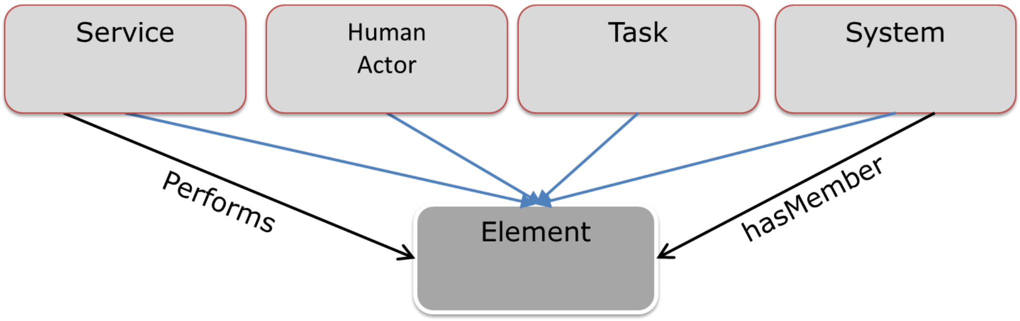

Figure 2: Service and elements of SOA

Figure 2 shows that the elements of SOA include Services, Human Actors, Tasks and Systems. Services are performed by any of these elements. Systems have members which can be any element, including services, human actors, tasks and systems themselves. Further explanation of tasks and systems are in the following clauses.

3.2.3 Semantics

Services in a service oriented architecture (SOA) should address more than syntactic interoperability. It is important to have an emphasis on application semantics as well as a syntactic connect via a defined interface, defined protocol, message format etc. The business user community needs to be able to satisfy its goals and objectives being met via services with a business emphasis. The services in SOA, need to be able to address business requirements beyond the syntax part or just the semantics of the mechanics part of the web services for messages and interfaces, by really addressing semantics of the business requirement---in particular the semantics of the community of concepts via the use of ontologies

3.2.4 Tasks and Activities

A *task* is an atomic action which accomplishes a defined result. Tasks are done by people or organizations, specifically by instances of Human Actor. Human tasks are tasks which are done by people or organizations, specifically instances of Human Actor. Because Tasks are atomic, a task is done by at most one instance of Human Actor.

The word “activity” in the ‘Service’ definition above is used in the general English language sense of the word, not in the process-specific sense of that same word (i.e., activities are not necessarily process activities). In particular, the Business Process Modelling Notation (BPMN) 2.0 defines *task* as follows: “A *task* is an atomic Activity within a Process flow. A task is used when the work in the process cannot be broken down to a finer level of detail. Generally, an end-user and/or applications are used to perform the task when it is executed.” This formally separates the notion of doing from the notion of performing. Tasks are (optionally) done by human actors, furthermore (as instances of Element) tasks can use services that are performed by technology components.

3.2.5 Compositions and Processes

A Composition is a system that is the result of assembling a collection of things for a particular purpose.

In this case composition to refer to something, not the act of composing.

Compositions are organized according to one of a set of patterns or styles, orchestration, choreography, and collaboration.

Just as systems are recursive, compositions are recursive, so that a composition can be part of another composition, but a particular composition cannot be a part of itself

Since a composition is a collection, it must use at least one other element and those elements can be outside its own boundary. For SOA, these elements are usually, but not limited to, services, other compositions, processes, actors, and tasks.

Compositions are not visible to external observers, like services. However, compositions patterns offer insight to the internal viewpoint of the composition and describe the way in which a collection of elements are assembled or used to achieve a result.

Services can represent or be implemented as a composition.

As shown in figure 3, there are two subclasses of compositions that are especially important for SOA, Service Compositions and Processes.

Patterns of composition

Compositions can be realized using 3 common patterns or styles which can be distinguished by the presence of a director of the composition and the existence of a predefined pattern of behaviour or flow.

- Orchestration is a pattern where there is one element used by the composition that oversees and directs the other elements. The directing element is different from the orchestration and maybe inside or outside the boundary. For example, a workflow is a member of the composition but is executing the flow.

- Choreography is a pattern for compositions where the elements used by the composition interact in a non directed fashion (no director) and every member knowing and following the pattern of behaviour. There is a predefined shared pattern or flow of behaviour.

- Collaboration is a pattern for compositions where the elements used by the composition in a non directed fashion (no director), and every member acts according to their own plan/purposed without a predefined pattern or flow of behaviour.   Interactions between members occur as needed by each member.

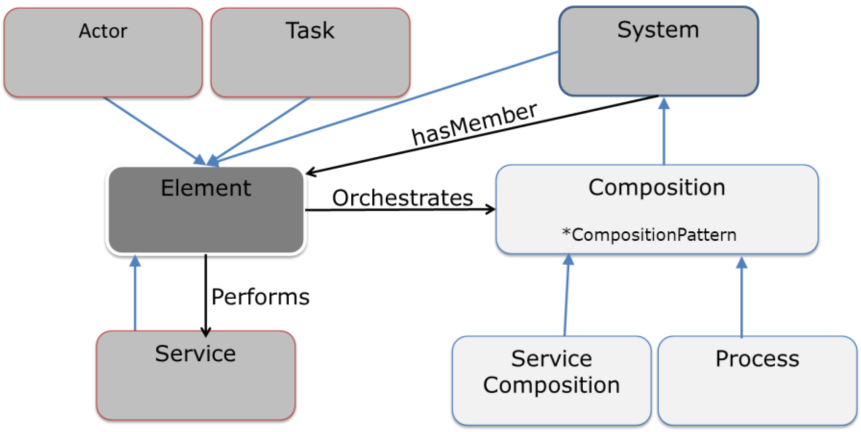

Figure 3: Composition and its sub-classes

Figure 3 ties together the concepts of service, system and composition. Elements of SOA systems are Services, Actors, Tasks, and Systems. Any of these elements can perform a service. A System can have any of these elements as members of the system. Compositions are systems and therefore can have members that are services, actors, tasks, and systems. Compositions have a property that indicates the composition pattern it supports, orchestrated, choreographed, or collaborative. Only compositions which use the orchestration pattern have an element that orchestrates the members of the composition. Service Compositions and Processes are Compositions; therefore they can also exhibit any of the composition patterns.

Service compositions

Service compositions are compositions that provide (in the operational sense) higher level services that are only composed of other services. Service compositions can exhibit one or more of the 3 composition patterns: orchestration, choreography, and collaboration.

Processes

Processes are compositions whose elements are composed into a sequence or flow of activities and interactions with the objective of carrying out work.

As shown in figure 3, Processes can be composed of elements like human actors, tasks, services, and processes. In addition, processes can exhibit any of the composition patterns.

"A process always adds logic via the composition pattern; the result is more than the parts. According to their collaboration pattern, processes can be:

- **Orchestrated**: When a process is orchestrated in a business process management system, then the resulting IT artifact is in fact an orchestration; i.e., it has an orchestration collaboration pattern. This type of process is often called a *process orchestration*.

- **Choreographed**: For example, a process model representing a defined pattern of behavior. This type of process is often called a *process choreography*.

- **Collaborative**: No (pre)defined pattern of behavior (model); the process represents observed (executed) behavior. " (Note: See Bibliography [21])

Business Process

A Business Process is a defined set of activities that represent the steps required to achieve a business objective. It includes the flow and use of information and resources.

- Business Process Management - Through robust and flexible software capabilities and industry expertise, business process management enables discovery, modelling, execution, rapid changing, governing, and gaining end-to-end visibility on business processes. (Note: see Bibliography [23])

- Business Activity Monitoring - Monitoring business operations and the processes and associating SLA, KPI (Key Performance Indicator) with the actual business processes to visualize the linkage.

3.2.6 Service Registration and Discovery

Service discovery is the process that service consumers use to search and retrieve desired services according to their specific functional or non-functional requirements. Service discovery is done during design time as well as during runtime. Discovery may require the ability to query for the service metadata.

Registration is the publication process by which metadata about services is stored in a registry, repository, or made available to the service discovery mechanism in some way. Registration may require the ability to add, delete, modify, classify and verify the metadata.

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

During runtime, usage of a service by a consumer involves two steps – service discovery and location, and service invocation. The consumers of services interact with the Service Registry (needed to support Governance as well) to find the service. The Service Container invokes the Service Component to perform 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.

Management of registration processes may require management of storage, backup and recovery, and notification.

The figure below show these steps.

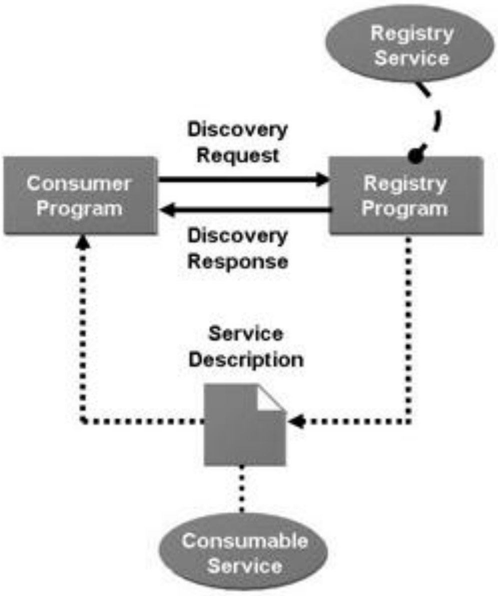

Figure 4 Model for Service Discovery

The consumer program can be a program that performs a service. It is a consumer of the registry service, and is also a consumer of whichever of the consumable services it uses.

Service Repository

Re-use of software services is a very important aspect of SOA for many enterprises. They often introduce governance mechanisms to ensure that services are developed with re-use in mind, and that the possibility of re-using existing services is explored before new ones are written. But, for re-use to be possible at all, the services must be clearly described, and their descriptions must be readily available to developers.

The simplest way of making service descriptions available is to publish them as documents. A more sophisticated mechanism, which has advantages where there are large numbers of services to keep track of, is a *service repository*, which enables you to maintain and search a database of service descriptions.

In a large enterprise you may have a very large number of services, and you will need a common vocabulary and data model, as well as some sort of knowledge management system, as the basis for your service repository.

As shown in figure 4, Consumers may need to discover and access service descriptions during design and development time. The metadata and interface description is essential to properly develop the consuming programs and solutions.

Service Registry

A *service registry* is an organized collection of service descriptions, maintained by a *registry service* that returns the descriptions that match specifications in submitted enquiries, as described in the Model for Service Discovery. A registry service has two principle interfaces:

- For management of the service descriptions that it maintains
- For enquiries and responses

The Universal Description Discovery and Integration (UDDI) standards published by OASIS are the most commonly-accepted standards for both of these interfaces. They apply to web-based registries that expose information about businesses or other entities.

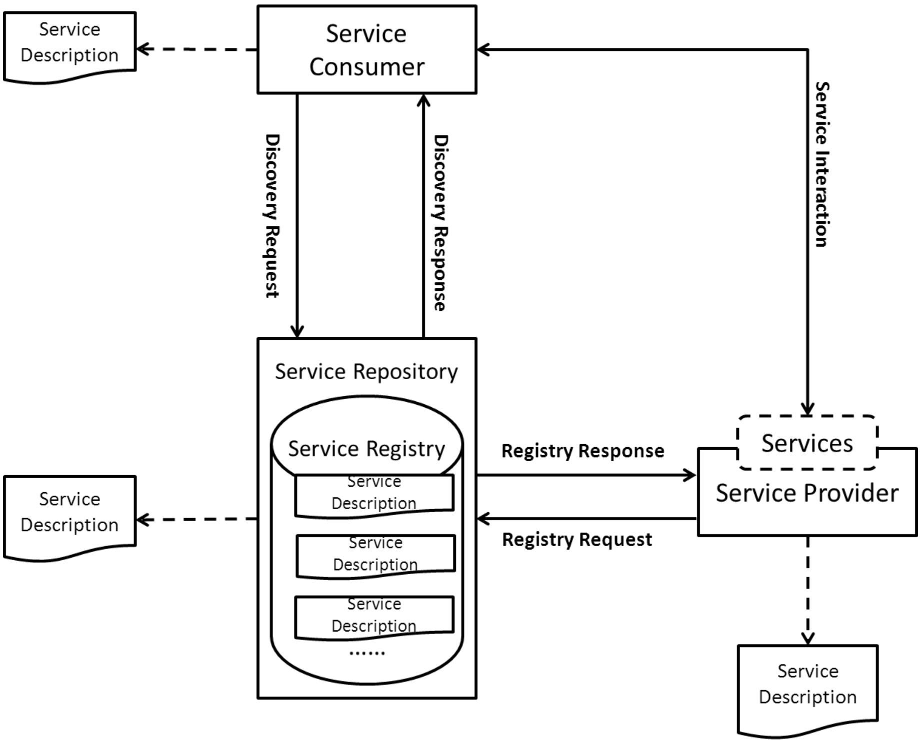

Figure 5: Example of a Service Registry as part of a Service Repository

As shown in figure 5, the service registry information may be part of the service repository applications, however, the interactions with the service registry are typically during runtime to find the address to interact with for the service and service provider.   Registry applications can also exist independently of the repository application.

3.2.7 Service Description, Interfaces, Contracts and Policies

Service descriptions can contain service interfaces, contracts and policies. Therefore they will be defined before defining service description.

Service Interfaces

Service interfaces define the way in which other elements can interact and exchange information with a service as the outcome of a request in the definition of a service (Note: See Bibliography [21]). It is important that services have simple, well-defined interfaces. This makes it easy to interact with them, and enables other elements to use them in a structured manner. The concept of an interface includes the notion that interfaces that define the parameters for information passing in and out of them when invoked. The specific nature of how an interface is invoked and how information is passed back and forth differs between domains. Service interfaces are typically, but not necessarily, message-based (to support loose-coupling). Furthermore, service interfaces are always defined independently from any particular service implementation (to support loose-coupling and service mediation).

Any service must have at least one service interface. There can be constraints on the allowed interaction on a service interface such as only certain value ranges allowed on given parameters. Depending on the nature of the service and the service interface in question, these constraints may be defined either formally or informally (the informal case being required at a minimum for certain types of real-world services).

The same service interface can be an interface of multiple services. This does not mean that these services are the same, nor even that they have the same effect; it only means that it is possible to interact with all of them in the manner defined by the service interface in question.

Service Contracts

A *service contract* defines the terms, conditions, and interaction rules that interacting participants must agree to (directly or indirectly). A service contract is binding on all participants in the interaction, including the service itself and the element that provides it for the interaction in question. Sometimes, specific agreements may be needed in order to define how to use a service in order to regulate its use or to ensure that the service interactions are in a certain sequence.

Service contracts explicitly regulate both the operational interaction aspects and the legal agreement aspects of using services, like Service-Level Agreements (SLAs). It is possible as an architectural convention to split the interaction and legal aspects into two different service contracts, depending on the needs of the application.

Anyone wanting to interact with a service must obey the interaction aspects of all service contracts that apply to that interaction. The operational interaction aspect of a service contract is different from the notion of a service interface in that a service contract does not define the service interfaces themselves, rather defines any relevant multi-interaction and/or sequencing constraints on how to use a service through interaction across its set of service interfaces. As a simple example a payment service may have a service contract stating that a payment must be created before it can be tracked.

In addition to the rules and regulations that intrinsically apply to any interaction with a service there may be additional legal agreements that apply to certain human actors and their use of services. The actual legal obligations on each of these human actors are defined in the service contract. This can include defining who is the provider and who is the consumer from a legal obligation perspective.

A given human actor may be party to none, one, or many service contracts. Similarly, a given service contract may involve none, one, or multiple human actors. Note that it is important to allow for sourcing contracts where there is a legal agreement between human actor A and human actor B (both of which are party to a service contract), yet human actor B has sourced the performing of the service to human actor C (*aka* human actor C performs the service in question, not human actor B).

Note that Service Contracts can (in their legal part) express temporal aspects such as “is obliged to at this instant”, “was obliged to”, and “may in future be obliged to”.

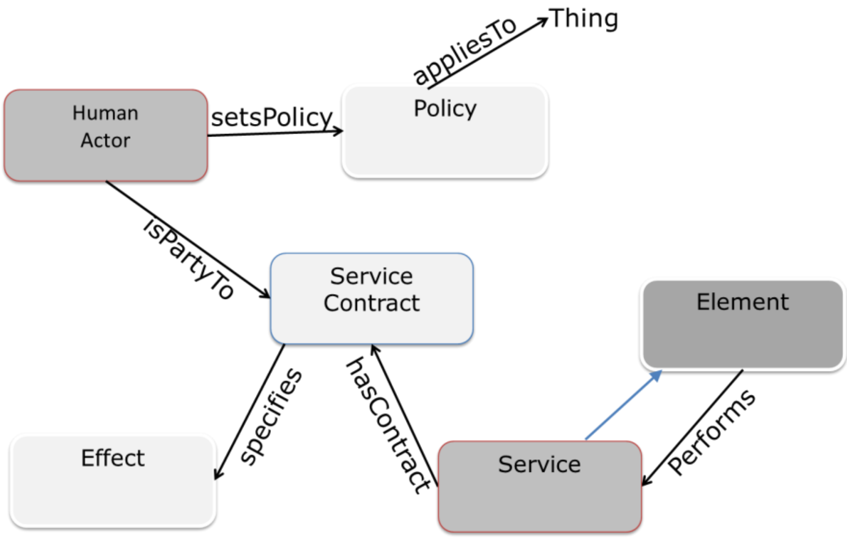

Figure 6: Contract and Policy

Figure 6 shows the relationship of service contracts with services and policies. Services have service contracts. Service contracts specify the effects of the services. Human Actors are parties to service contracts and set the policies that apply to any element in the SOA solution. Service descriptions and policies can be referred to by the service contract.

Policy

A *policy* is a statement of direction that a human actor may intend to follow or may intend that another human actor should follow. Knowing the policies that apply to something makes it easier and more transparent to interact with it. Policies, the human actors defining them, and the things that they apply to are important aspects of any system, certainly also SOA systems with their many different interacting elements. Policies can apply to any element in a system.

.

From a design perspective, policies may have more granular parts or may be expressed and made operational through specific rules.

Policies are different from Service Contracts. While policies may apply to service contracts – such as security policies on who may change a given service contract – or conversely be referred to by service contracts as part of the terms, conditions, and interaction rules that interacting participants must agree to, service contracts are themselves not policies as they do not describe an intended course of action.

Policy as a concept is generic and has relevance outside the domain of SOA. Policies can apply to things other than elements; in fact, policies can apply to anything at all, including other policies.

Policies can apply to zero (in the case where a policy has been formulated but not yet explicitly applied to anything), one, or more things. Note that having a policy apply to multiple things does not mean that these things are the same, only that they are (partly) regulated by the same intent. In the other direction, things may be subject to zero, one, or more policies. Note that where multiple policies apply to the same thing this is often because the multiple policies are from multiple different policy domains (such as security and governance).

Policies can be set by zero (in the case where actors setting the policy by choice are not defined or captured), one, or more human actors. Note specifically that some policies are set by multiple human actors in conjunction, meaning that all these human actors need to discuss and agree on the policy before it can take effect.

Service Description

A service description contains the information necessary to interact with the service and describes this in such terms as the service interface, (service inputs, outputs, and associated semantics), the service contract. (what is accomplished when the service is invoked and how to accomplish such effect), and service policies.(conditions for using the service).

The purpose of combining service interfaces, service contracts and service policies in a composite service description is to facilitate interaction and visibility, particularly when the participants are in different ownership domains (Note: See Bibliography [19]) for an explanation of Ownership Domain). Providing explicit service descriptions makes it possible for potential participants to construct systems according to the descriptions of services desired to be used, rather than according to implementations of said services.

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

3.2.8 Service Lifecycle

The service lifecycle consists of the activities, roles and work products for a service throughout its life, from identification to instantiation and retirement. These are often an extension of the organization's software development lifecycle.

A service lifecycle includes Service Definition, Service Realization Planning, Service Modelling, Service Implementation, Assembly, or Acquisition, Service Testing, Service Deployment, Service Management and Monitoring, Service Support, and Service Retirement.

The full service lifecycle should be managed and governed, which includes Service Governance, Policy Management, Requirements Management, and Configuration Management. Asset and registry service implementations are commonly used to manage and govern since they provide access to some of the portfolio of assets necessary to support the lifecycle and its management. These assets include service implementations, processes, documents, etc.

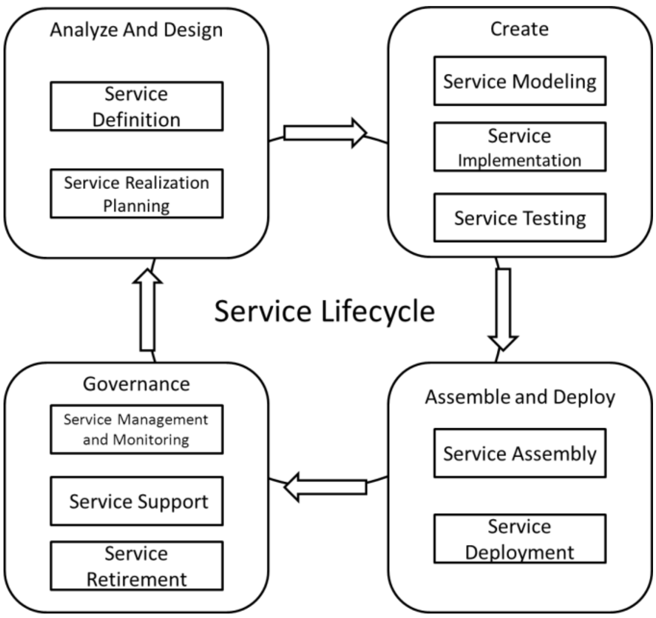

Figure 7 Example of a Service Lifecycle

3.2.9 SOA Lifecycle

The SOA lifecycle describes the activities, roles, and work products as they relate to the modelling of SOA solutions throughout the lifecycle, from identification to retirement. A SOA lifecycle is where the services and business process is modeled, assembled, deployed and monitored in an iterative manner in order to provide business solutions based on SOA.

The SOA lifecycle begins with modeLling the business (capturing the business design) including the key performance indicators of the business goals and objectives, assembling and translating that model into an information system design, deploying that information system, managing that deployment, and using the results coming out of that environment to identify ways to refine the business design. It is a premise of the lifecycle that feedback is cycled to and from phases in iterative steps of refinement and that the model may actually be built using reverse-engineering techniques or other means to facilitate the needs of the business.

The lifecycle is then layered on a backdrop of a set of governance processes that ensure that compliance and operational polices are enforced, and that change occurs in a controlled fashion and with appropriate authority as envisioned by the business design.

- **Model** - Modelling is the process of capturing business design from an understanding of business requirements and objectives and translating that into a specification of business processes, goals and assumptions – creating an encoded *model* of the business, along with key performance indicators.
- **Assemble –** Assemble uses the business design to *assemble* the information system artifacts that will implement the business design. The business design is converted into a set of business process definitions and activities deriving the required services and from the activity definitions.
- **Deploy -** Deploy includes a combination of creating the hosting environment for applications and the actual deployment of those applications. This includes resolving the application's resource dependencies, operational conditions, capacity requirements, and integrity and access constraints. In addition, the techniques you will employ for ensuring availability, reliability, integrity, efficiency, and service ability should be considered.
- **Manage -** Manage considers how to maintain the operational environment and the policies expressed in the assembly of the SOA applications deployed to that environment. This includes monitoring performance of service requests and timeliness of service responses; maintaining problem logs to detect failures in various system components; detecting and localizing those failures; routing work around them; recovering work affected by those failures; correcting problems; and restoring the operational state of the system.

The Lifecycle Flow - Progression through the lifecycle is not entirely linear. In fact, changes to key performance information in the Model phase often need to be fed directly in to the Management phase to update the operational environment. Constraints in the Deploy phase, such as limiting assumptions about where resources are located in the system, may condition some of the Assembly phase decisions. And, occasionally, information technology constraints established in the Assembly phase will limit the business design created during the Model phase – for example, the cost of wide-area wireless communication with remote hand held devices may be prohibitive to deploying a field force to rural locations and therefore needs to be reflected back into the business design.

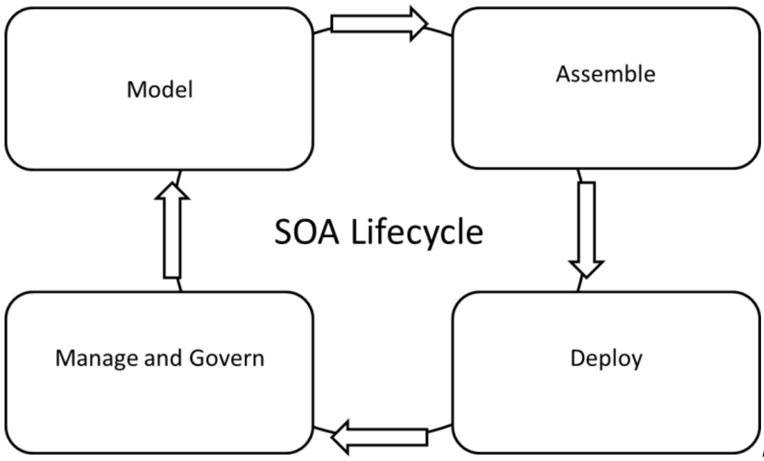

Figure 8: SOA Lifecycle

3.3 Cross Cutting Aspects

Cross cutting aspects are capabilities and functionality that are valid for and apply to multiple roles or functional layers. For example management is a cross cutting aspects because it applies to operational systems, services, business processes and consumers. All of these need to be managed, but how these are managed is different based on the management target. So, managing infrastructure like storage and data bases is very different from managing business processes. Cross cutting aspects can apply to other cross cutting aspects, so governance applies to the functional layers as well as integration, information and management.

Critical success factors for a deployable SOA architecture are:

3.3.1 Integration

Over the past years we've seen a rapid adoption of the SOA architectural style as the underpinning of enterprise architectures. Many enterprises are now moving beyond SOA projects that focused on specific, departmental business problems to more complex SOA installations extending the reach of SOA and making it ubiquitous, supporting end-to-end business interactions across business unit boundaries and partners.

As these enterprises move from departmental to enterprise, they can find themselves structured as a set of autonomous service domains, which now need to be federated and have some aspects managed globally. From a business perspective, this really means scaling up or extending service reuse, so that existing and new services can be reused across some subset of the service domains in the enterprise. This clearly shows the value of being capable of interacting and managing across service domains

Integration aspects for SOA solutions are central to developing SOA solutions because most of the time there are cross service and solution domain interactions between the services in a SOA solution. Integration aspects and supporting technologies need to be identified even when service interactions are simpler and within a domain.

Cross Domain interaction:

Service discovery and management may need to span various solution domains in an enterprise or across enterprises. In addition, discovery and management may have to work across different technologies since different solution domains may have selected different means for their SOA implementation.

The figure 9 describes shows a number of solution domains, each implemented with its own technology, where the interactions between them need to be managed as a whole.

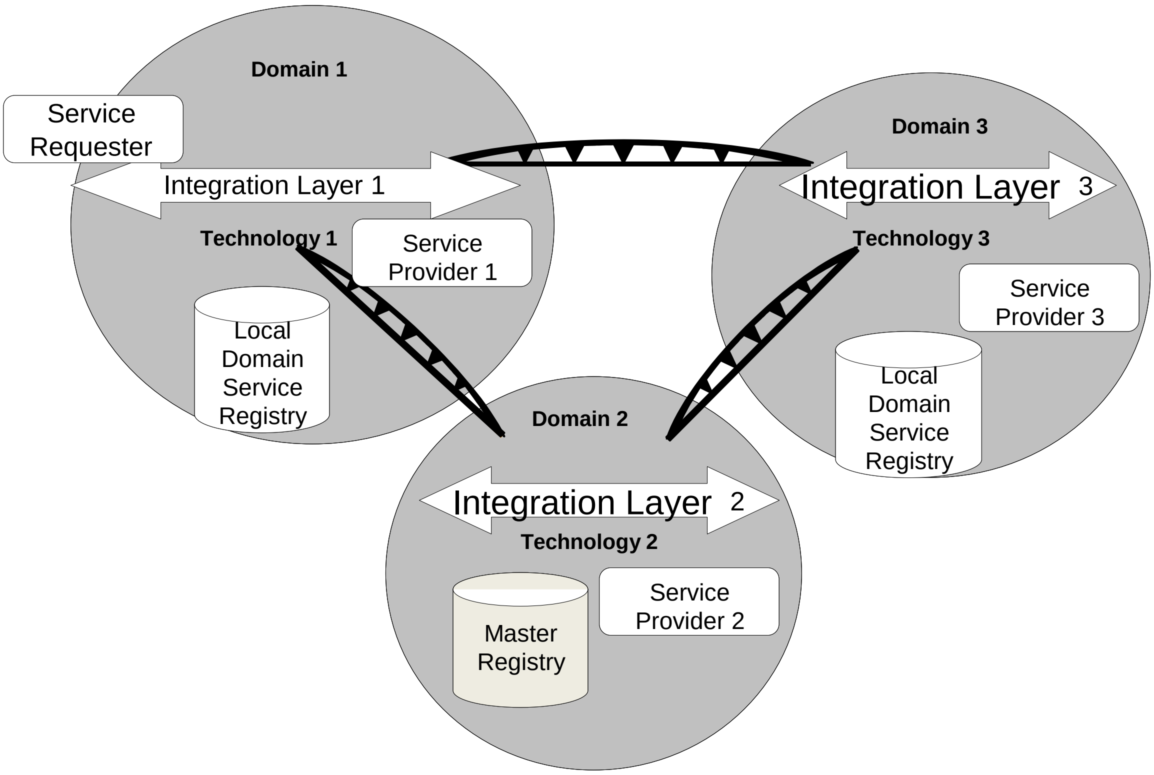

Figure 9 Cross Domain Interaction

Both single solution domain SOA infrastructure and heterogeneous federated SOA infrastructure must address the management of cross cutting aspects, especially of service discovery, service management, service security and service governance (life cycle control).

Depending on the service consumer's technical or business context the same service may be available from various providers both inside and outside the enterprise as well as through various technologies. These services have concrete endpoints that may reside in different service domains. An integration layer, and associated registry, in the consumer's service domain makes finding and using (binding to) these different service implementations simpler.

Service Integration

Service integration is crucial as it provides the capability to mediate, transform, route and transport service requests from the service requester to the correct service implementation. This layer may include an intelligent routing, protocol switching, and interface transformation mechanisms as is often seen in an Enterprise Service Bus (ESB). The integration layer 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 layer is not only a technical loose coupling addressing protocols, locations or platforms, but can also be a business semantic loose coupling performing required adaptations between service requester and provider.

The implication of loose coupling and de-facto heterogeneity of the enterprises is the need for a stronger management of the services and the corresponding artifacts.

Three aspects of integration that need to be considered are transportation, transformation and mediation.

Transport

In order for two ends to communicate they need to be connected to a transport infrastructure, that provides the protocols necessary to convey messages between the two. If there is no common transport infrastructure between the two then will be necessary to bridge between different infrastructures. This may require protocol translations, as well as the need to provide routing and correlation of messages between the end points.

Transformation

Sometimes two end points might not share the same definition of message content, so even though they can exchange protocol messages, the content being exchanged might have to be transformed into a format that the other side accepts. An example here is where one end-point uses a different version of a data format with extra fields, so the extra fields need be removed in order for the message be acceptable to the other side.

Mediation

When integrating processes it is not always the case that a message on one side equates to a single message on the other. The two sides might be at different levels of abstraction with one message resulting in the fan out of multiple messages, with a need to combine responses into a single reply. Mediation can combine transport and transformation integration with these more complex process integration patterns.

3.3.2 Management and Security

This aspect supports non-functional requirements (NFR) related as a key feature/concern of SOA and provides a focal point for dealing with them in any given solution. It provides the means of ensuring that a SOA meets its requirements with respect to: monitoring, reliability, availability, manageability, transactionality, maintainability, scalability, security, safety, life cycle, etc. It has the same scope as the traditional FCAPS (Fault, Configuration, Accounting, Performance, Security) from ITIL or RAS (Reliability, Availability, Serviceability). Three important aspects of QOS that need to be supported are Transaction management, availability and service reliability.

Transaction

Transactions coordinate the service's actions on resources that can belong to distributed components when these components are required to reach a consistent agreement on the outcome of the service's interactions. A transaction manager usually coordinates the outcome while hiding the specifics of each coordinated service component and the technology of the service implementation. With that approach a common and standardized transactional semantic can be used on the services.

The transactions consist of two types of transactions that are both covered by Web services standards i.e. WS-Coordination (Note: See Bibliography [34]) which is the framework for WS-AtomicTransaction (Note: See Bibliography [35]), and WS-BusinessActivity (Note: See Bibliography [36]).

Atomic transaction
Atomic transactions occur when building services components that require consistent agreement on the outcome of short-lived distributed activities that have the all-or-nothing property. Atomic transaction require Atomicity, Consistency, Isolation and Durability (ACID) properties and thus maintain a lock on the changed resources until these changes are committed or rolled back. The outcome of a service will be either a successful change of all the resources coordinated by the service, or the restoration of the state that existed before the service interaction.

Business activity transactions
Business activity transactions occur when building choreographies that require consistent agreement on the outcome of coordinated long-running distributed services interactions where locks cannot be maintained on

modified resources for the time scale of the choreography. Because of the usual stateless aspects of services the changes that occurred on the resources managed by a service component cannot be rolled back after the service interaction. Therefore, in case of failure of the coordination, the opposite actions must be taken through appropriate services interactions to restore an acceptable state for the resources that are controlled by each service component. These actions are called compensation and may still lead to permanent changes. As an example some services may not allow the deletion of a newly created resource, in which case the compensation action might be the setting of a status to inactive. Business activity transactions can be recursive in nature which would incorporate multiple business activity scopes.

Availability

The availability of a SOA solution is the percentage of time that the solution is performing its business functions appropriately. In order to measure this, it is important to identify and monitor key performance indicators for the solution as a whole. Measuring individual service availability will not reflect the availability and execution of the business functions.

Availability of services is tied to the percentage of time that a service can be successfully invoked. Availability of services can be measured from a provider, network, and consumer boundary, checking that the service, system, and network are functioning. Service availability is different from different boundaries, i.e. sometimes the provider’s service, system and network is working fine but there is a bottle neck in the network between the consumer and the provider. Often service availability can be measured by infrastructure and tools on behalf of the service and metrics provided are common, i.e. percentage of up time and number of failed messages.

Service availability is frequently a key metric in service level agreements, policies and contracts.

Service Reliability

Reliability in the context if SOA is a quality of service directed at the interactions between a service consumer and a service provider, and is used to indicate differing levels of assurances on the intermediaries and network involved in the exchange of messages during service invocations. Reliability is an end-to-end property, meaning that the same quality should be preserved throughout the whole path from consumer to provider.

On key principle when designing a service is the concept of idempotency. If a service is idempotent then duplicates of the same service invocation, for example through retries, will result in exactly the same result. In other words an identical result is achieved whether the request happens once or is repeated several times. All non state changing (read/get) operations are by definition idempotent. To make a state changing operation idempotent requires careful design to ensure that duplicate requests result in the same state. Where is it not possible to design an idempotent service, other reliability qualities may be requested to help, notably At-most-once, and Exactly-once.

Four reliability qualities may be provided by SOA systems:

- **At-least-once:** a request is delivered once, but it is ok to deliver duplicates. Idempotent operations are well suited to these semantics.

- **At-most-once**: a request is delivered once and it is not allowed to deliver any duplicates as the effect of a duplicate request would result in incorrect behaviour for example, deducting the same money from a bank account! The correct behaviour of the service is not at risk if the request fails to get delivered.

- **Exactly-once:** a request is delivered once and it is not allowed to once deliver any duplicates. The correct behaviour of the service may be at risk if the request fails. In other words all reasonable attempts should be made to deliver the request once and only once.

- **Ordered:** A service consumer may initiate multiple different requests to the same provider over a period of time. since underlying intermediaries and networks might get such requests out of order, if it is important that the order is preserved in the request ordering if it is required between consumer and provider. Note that ordering can be combined with the three other properties.

Reliability contrasts with availability and transactionality. While Availability concerns itself with whether a service is active, up and running, and able to process requests or not, reliability concerns itself with ensuring the request and response messages are delivered. On the other hand, reliability concerns itself with message delivery between consumer and provider, it does not address whether a request is actually processed or not by the provider. Transactions address whether a request is process or not.

SOA Security

SOA security addresses the protection against threats across the vulnerability dimensions of a service oriented architecture, this protecting the interactions between service consumers and service providers, but also protecting all of the elements that contribute to the architecture.

The approach used for SOA security follows the industry best practices such as the recommendation developed by ITU-T on X.805 Security architecture for systems providing end‑ to‑ end communications and the dimensions covered by the WS-Security set of standards.

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

The eight security dimensions that are required to protect against the threats listed above are the following:

1. Access Control: Limit and control access to services and elements using control tokens, components and elements: password, Access Control Lists, firewall.
2. Authentication: Provide a proof of identity for the use of services or any components in the SOA architecture using for examples: shared secret, PKI, digital signatures, digital certificates. It is to be noted that as SOA addresses interactions between service consumers and providers the proof of identity should be carried between the initial consumer and ultimate provider. In service choreographies there will be multiple actors interacting which may all have to be authenticated. There aspect that SOA have to take care of is that when services are offered between to entities that are organizations it may become too complex to manage the churn of actors joining or leaving these entities. This is why federated identities that implement a trustee model between entities are used across SOA, with the propagation of assertion tokens that provide authentication of the initial requester through a trustee protocol between components. An example implementation is WS-Federation.
3. Non-repudiation: Prevent ability to deny that an activity on components or elements in the Service Oriented Architecture occurred. Examples of proofs: system logs, digital signatures protecting messages with XML-Signature. In service choreographies there will be multiple actors interacting and it may become necessary to maintain a trace of all actions from all actors.
4. Data Confidentiality: Ensure confidentiality of data exchanged in services interactions or managed by components of the architecture. The confidentiality may have to be carried with no disruption should be

carried between the initial service consumer and ultimate service provider with no disruptions to avoid unnecessary disclosures. Example protecting messages using encryption with XML-Encryption.

5. Communication Security: Ensure information only flows from expected source to expected destination. Examples: using HTTPS, VPN, MPLS, L2TP. In an SOA architecture an application level validation on component boundaries may become necessary to control incoming and outgoing sources and destinations.

6. Data Integrity: Ensure that data is received as sent or retrieved as stored in all of the interactions between service components, whether they are for SOA management or lifecycle purposes or for interactions between service consumers and providers. Examples: digital signature with XML-Signature, anti-virus software. The XML signature should only be processed at both end to avoid opening a breach in the various nodes that constitute an SOA architecture between the service consumer and provider.

7. Availability: Ensure that elements, services and components are available to legitimate users. This has to be coordinated with the SLAs , and other contract metadata that specify the conditions of use of the service.

8. Privacy: Ensure identification, service use and service management are kept private.

SOA Security Governance

SOA security governance is a specific domain of governance addresses the SOA security.

- SOA Security functions: SOA security governance controls the lifecycle of the components that provide functions for the eight security dimensions to protect from the five threats as listed before.
- Security Policy Infrastructure: SOA security governance defines and implements the components that administer security policies, the security policies distribution and transformation to the appropriate SOA components and elements, the security policy decision and enforcement components and finally the components that monitor and report on security policies use and conformance.
- SOA Security processes: SOA security governance defines the IT and business processes for risk and compliance, trust management, identity and access control lifecycle, data protection and disclosure control, and finally security for all components in a service oriented architecture such as registries, ESB etc. In a multi entity environment these process may include the definition and exchange of certificates or public keys between trusted actors before enabling other actors in the entities to interact.

It is to be noted that confidentiality and non-repudiation may cause augmentation of data size because of required additional information such as certificates and additional processing time to control signatures.

SOA Management

The same kinds of management that apply to business in general 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. While Service Management is focused on the management of services, SOA management is has a broader scope and manages the entire SOA solution, or set of SOA solutions. Common types of management used for SOA solutions are:

- **IT Systems Monitoring and Management:** Provides monitoring and management of IT infrastructure and systems, including the ability to monitor and capture metric and status of IT systems and infrastructure. This also includes management of virtualized systems.

- **Application and SOA Monitoring and Management:** Provides monitoring and management of software services and application, this includes ability to capture metrics and to monitor and manage application and solution status.
- **Business Activity Monitoring and Management:** Provides monitoring and management of business activities and business processes. It provides ability to analyze this event information, both in real-time / near real-time, as well as stored (warehoused) events and to review and assess business activities in the form of event information and determines responses or issues alerts/ notifications.
- **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, 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.
- **Event Management:** Provides the ability to manage events and enables event processing, logging, and auditing.
- **Configuration and Change Management:** This category of capabilities provides ability to change solution and service configuration and descriptions.
- **Policy Monitoring and Enforcement:** Provides mechanism to monitor and enforce of policies and business rules for the SOA solution and services. This includes finding and accessing policies, evaluating and enforcing policies at check points. Policy enforcement includes enforcement when metrics are captured, signalled and recorded. Enforcement must also send compliance status or metrics, as well as notification and log of non-compliance
- **Lifecycle management:** Provides a mechanism to deploy, start/enable, stop/disable, and undeploy services and SOA solutions.

In addition, SOA and service governance will use management to actually execute on and enforce the governance policies and processes. These governance policies along with the other polices in the system for security, reliability, availability, etc. are the rules that drive the management systems. (Note: See Bibliography [20])

Management Services

Management services represents 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. Since the business design can be captured as a model, and used that to assemble the application services that implement that design, correlation between the business and the IT system can be captured. This correlation, if carried into the deployment environment can be used by management services 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 can be used in the service models as part of the SOA solution to both enable and manage the functional services and SOA solutions.

Management services are different from the functional interfaces and the management interfaces implemented directly by services to enable manageability. (Note: See Bibliography [20])

Service Metadata Management

In SOA environments, the capability to manage additional service metadata and physical documents such as the service descriptions and policies related to the services becomes essential. Registries and repositories enable this capability. It is important to have this in addition to service discovery.

A set of service information is necessary to enable management and increase loose coupling. This set of information also enables the capability of using registries, repositories, integration layers, and cross domain federation.

The information needed in registries can include Physical Documents that describe a service, Logical Derivations for communications to be enabled, and Service Related Entities to maintain relationships to potentially impacted entities

Metadata for service descriptions that needs to be accessible can include: Service properties (other information about the service), Service relationships (information about the relationships with the service), and Classification of services (for matchmaking)

There are other descriptions that may be needed as well for understanding service domains, the integration layer (or ESB), and service federation.

3.3.3 SOA Governance

In general, governance means establishing and enforcing how people and solutions work together to achieve organizational objectives. This focus on putting controls in place distinguishes governance from day to day management activities (Note: See Bibliography [22]).

Whilst governance has been around a long time, SOA has heightened the need and importance of having a formal SOA Governance Regimen that sets expectations and eases the transition of an organization to SOA by providing a means to reduce risk, maintain business alignment, and show business value of SOA investments through a combination of people, process, and technology. The role of the SOA Governance Regimen is to create a consistent approach across processes, standards, policies, and guidelines while putting compliance mechanisms in place.

This clause is sourced from The Open Group SOA Governance Framework Technical Standard (Note: See Bibliography [22]).

SOA Governance should be viewed as the application of Business Governance, IT Governance, and EA Governance to Service-Oriented Architecture (SOA). In effect, SOA Governance extends IT and EA Governance, ensuring that the benefits that SOA extols are met. This requires governing not only the execution aspects of SOA, but also the strategic planning activities. Many organizations already have a governance regimen for their IT department covering project funding, development, and maintenance activities. These can be defined using either one of the formal standard IT Governance frameworks – such as COBIT, ITIL, etc. – or an in-house governance framework that has been built over many years.

- **Enterprise Architecture (EA) Governance** is the practice and orientation by which enterprise architectures and other architectures are managed and controlled at an enterprise-wide level. (Note: See Bibliography [25])
- **Governance of IT** the system by which the current and future use of IT is directed and controlled. Corporate governance of IT involves evaluating and directing the use of IT to support the organization and monitoring this use to achieve plans. It includes the strategy and policies for using IT within the organization. (Note: see ISO/IEC 38500:2008 1.6.3)
- **Business Governance** is the set of processes, customs, policies, laws, and institutions affecting the way a organization is directed, administered, or controlled.

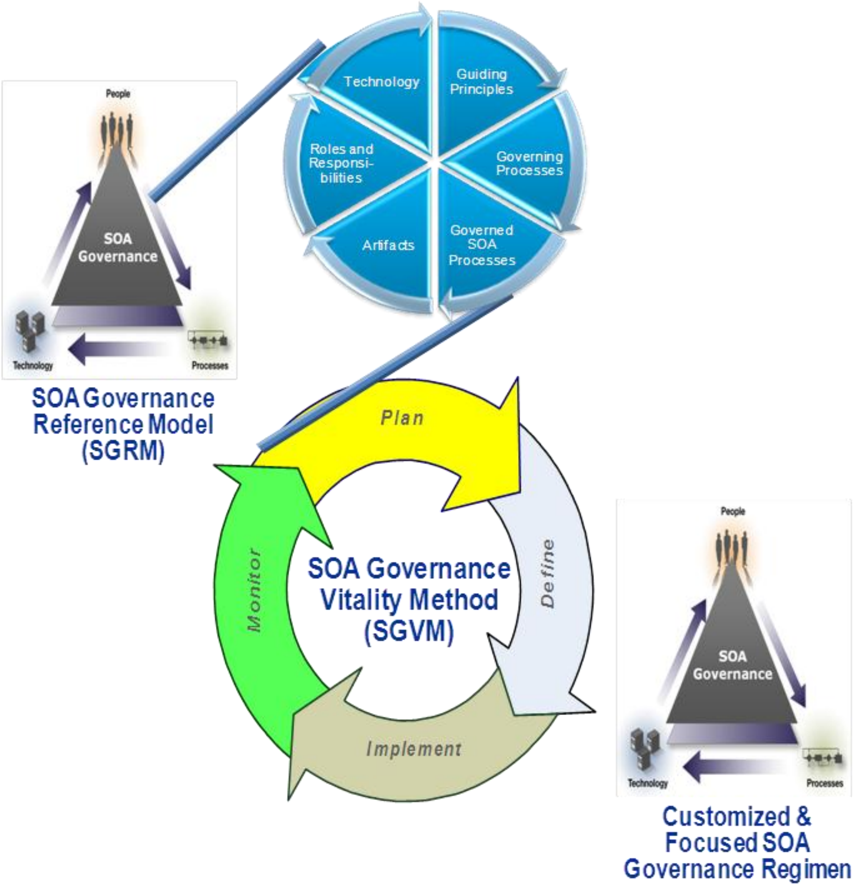

Figure 10 SOA Governance Regimen

Governance regimen for SOA solutions must include governance of services as well as governance of the SOA solution in its entirety. Governance must be applied to:

- **Processes** – including governing and Governed Processes
- **Organizational structures** – including roles and responsibilities
- **Enabling technologies** – including tools and infrastructure

The SOA Governance Framework can enable enterprises to define and deploy their own focused and customized SOA governance regimens, based on enterprise-specific factors such as business model, SOA maturity, and company size. It describes what decisions must be made, who should make them, how they are made and monitored, and what organization and tools will support them. It uses an incremental approach so that enterprises can continue to meet their current SOA demands while fulfilling their long-term aspirations and goals for SOA.
In order to specify the changes necessary to accommodate SOA in an existing governance regime, these governance activities must be mapped and integrated to the activities being utilized in the existing regime. While no two governance regimens are alike, governance standards can define best practices and provide a starting point for customization to the SOA solution.The SOA Governance Reference Model can be defined and used to develop an enterprise's customized SOA governance regimen. It should include governance activities that are impacted by SOA:

Guidelines – A set of guiding principles to inform decision-making in the design, deployment, and execution of its SOA solution and SOA governance regimen.

Governed Processes - Descriptions of the activities and processes that are subject to SOA governance. Four groups of activities are specific to the planning, design, and operational aspects of SOA:

- *SOA Solution Portfolio Management* determines which SOA solutions are developed and maintained.
- *Service Portfolio Management* determines which services are developed and maintained.
- *SOA Solution Lifecycle Management* is responsible for the development, operation, modification, and eventual withdrawal of SOA solutions. It raises requirements for service development that are addressed by service portfolio management, and requirements for service modification that are addressed by service lifecycle management.
- *Service Lifecycle Management* is responsible for the development, operation, modification, and eventual withdrawal of services.

Governing Processes - Descriptions of governing processes that control these activities. Governing processes realize the governance intentions of the organization. In general, they will be integrated with the existing enterprise governance processes, rather than forming a separate special set of processes for SOA. Three types of Governing processes are:

- The ***Compliance*** process ensures that the governed activities are carried out correctly by reviewing their outputs at the governance checkpoints.
- The ***Dispensations*** process allows a project or application team to obtain exceptions from the requirements of the compliance process in particular cases where blanket application of general policy is not appropriate.
- The ***Communications*** process is aimed at educating the people that take part in the governed activities, and communicating to them the systems architecture and the governance regimen. This includes setting up environments and tools to allow easy access to and use of governance information.

Roles and Responsibilities - An analysis of the kinds of people involved in the activities being governed, and their degrees of responsibility for those activities.

Artifacts - A set of artifacts to be used by the governing processes and should be kept current and available to governance stakeholders

Enabling Technology - Technology for enabling and implementing governing processes. A framework for technology capabilities can range in ability from manual processes to sophisticated software. Sometimes, the same technology used to implement the SOA solution can also be used to support the governance of it.

Since governance is about maintaining alignment, it requires an on-going process to support it. It is also unrealistic to expect an enterprise to define and deploy in a single project an enterprise-wide SOA governance regimen that meets its long-term aspirations and goals for SOA, particularly while continuing to meet its current SOA demands. A SOA Governance Vitality Method therefore needs to define and deploy SOA governance regimens incrementally. It takes the SOA Governance Reference Model as a baseline and tailors it to fit a particular enterprise through a number of phased activities that form a continuous improvement loop, defining a roadmap for deploying governance. It is not a one-shot activity but a continuous process in which progress is measured, course-correction is made, and updates are performed. The phases in such a method should include:

- Plan: identifies the requirements for SOA governance, reviews the existing governance regimen, and defines the vision, scope, and strategy for the changes to be made.
- Define: creates a tailored SOA governance regimen from the SOA Governance Reference Model, using the results of the Plan phase. It then analyzes the differences between this regimen and the existing governance regimen to create transition plans.
- Implement: realizes the tailored SOA governance regimen that was created in the Define phase to produce an SOA governance regimen for the enterprise; essentially it executes the transition plans from the define phase.

- Monitor: monitors the effectiveness of the operation of the SOA governance regimen that was produced in the Implement phase and whether it is meeting its intended purpose. This generates requirements for change that can be addressed in a new cycle of the SOA Governance Vitality Method.

Facilitated by:
- Repository
- Communications
- Centers of Excellence, Governance Boards

Benefits of:
- Business IT alignment
- Risk - Extending these existing governance models reduces the risk that organizations will create uncoordinated silo'ed governance regimens that will potentially duplicate existing coverage areas of their core governance regimens. It requires governing the strategic planning activities as well as the execution aspects of SOA. (Note: See Bibliography [22])

4 SOA Principles, Meta model, Capabilities and Assumptions

4.1 Architectural Principles

4.1.1 Architectural Principles defined

The architectural principles in SOA are driven to a degree by the importance of the 3 independence principles for SOA: location independence, implementation independence and protocol independence:

- Location independence: there are no preferred locations for a service consumers and service providers. They could transparently both be located on the same system, or in different organization in different physical locations.

- Implementation independence: there are no requirements for specific platform or implementation technologies for service consumers and service providers to adopt. They should not be aware of the other parties technical environment or implementation details to inter operate.

- Protocol independence: services can be exposed and consumed with a variety of transport protocols and message protocols. There may be a matchmaking protocol or component in the middle for interoperability purposes. In case of different protocols Enterprise Service Buses may perform the connection between heterogeneous services, otherwise there can be an agreement to use an interoperability profile such as defined by WS-I.In order to consume a service.

Services need to be enabled to be: interoperable, described, reusable, discoverable, composable, autonomous, loosely coupled, and manageable as described in the rest of this clause.

4.1.2 Interoperable – syntactic, semantic

The term "interoperability" is often used in a technical sense for example, systems engineering, or, the term is used in a broad sense, taking into account social, political, and organizational factors that impact system to system performance.

Syntactic Interoperability is when two or more systems are capable of communicating and exchanging data using some specified data formats or communication protocols, e.g. XML (eXtended Markup Language) or SQL (Structured Query Language) standards---much like interoperability of rail, which have common parameters like gauge, couplings, brakes, signalling, communications, loading gauge, operating rules etc. Syntactic interoperability is a .pre-requisite to achieving further interoperability.

Semantic interoperability is the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the end users of participating systems, all the participating sides must defer to a common information exchange reference model. The content of the information exchange requests are unambiguously defined: what is sent is the same as what is understood.

Schema:
"Any meaningful exchange of utterances depends upon the prior existence of an agreed set of semantic and syntactic rules. The recipients of the utterances must use only these rules to interpret the received utterances, if it is to mean the same as that which was meant by the utterer." (Note: See Bibliography Reference [24])

Facilitated by:

- Protocol Definition that includes:
- Definition of the messages types and content types to be exchanged (syntactic)
- Description of the content types, including any constraints on data values (to aid semantics)
- Definition on the underlying transport protocols that can be used (bindings)
- Definition of message exchange patterns, such as request and response messages
- Definition of failure modes, messages and states
- Description of any conversational or interactions e.g. call backs
- Definitions of any restrictions on invocation order e.g. open before write

Policies

- Polices that effect reliability, availability, security and transactionality

Benefits are:

- Increased understanding of the semantics of the service being provided
- Increased likelihood of two systems interacting correctly
- Decrease in erroneous interactions
- Reduction in ambiguity when errors and faults occur
- Less reliance on human misinterpretation of semantics

4.1.3 Described

The service description, as described in clause 4.2.6, is metadata that may include details on the, service contract, service interface and policies for both consumers and providers. The principle of being described is achieved when the service consumer, provider, or developer can find, access, interpret, and process a description of a service.

Facilitated by:

- Capturing the service interaction contract in a standardized format (i.e. XML, WSDL (Web Services Definition Language, Note: See Bibliography [27]) for Web services implementations)
- Storing the descriptions in the registry and/or repository so that it can be located, accessed and reused
- Isolating configuration into policy descriptions
- Capturing service level agreements in descriptions
- Capturing QOS available in descriptions
- Capturing business, governance, and management processes that use the services in descriptions
- Capturing service location

Benefits are:

- Enables loose coupling and late binding by providing sufficient description to be able to look up and bind to the service instance at runtime
- Enables reuse by adding metadata so that the appropriate services can be located and reused
- Enables reuse by adding description so that the appropriateness of the services available for the functions needed
- Enables management and reconfiguration of services at runtime and when being reused to minimize changes to the service implementation.
- Enables management of services at runtime by describing management monitoring and processes
- Enables business and IT governance of SOA solutions and services

- Enables addition of an integration layer or enterprise service bus (ESB)

4.1.4 Reusable

The principle of reusability is achieved when the same service definitions and/or implementations is used in more than one SOA solution. Enabling reusability helps reduce costs of the development and increase the return on investment for the solution long term.

Facilitated by:

- Service modelling and analysis – modelling the services to be developed to support the SOA solution, looking for common tasks and fine grained services in the solution
- Using the right granularity – finer grained 'utility' services may be reusable in service compositions
- Using autonomous definitions
- Adhering to principles for loosely coupled services

- Developing a well described service so that the service can be found, evaluated for appropriateness and reused.
- Making services discoverable
- Late binding of services, obtaining addresses for current service endpoint at runtime so that other services can be substituted at runtime and without redeveloping the service implementation
- Governance of the business with processes in place to identify redevelopment and require reuse of existing services
- Understanding how the service will be funded for development if it used across organizations
- Having the appropriate software development and maintenance lifecycle to support reuse of services in multiple solutions (i.e. impact when the implementation is updated)
- *Enabling management of services to be policy driven*
- *Implementing services with configuration isolated into policy*

Benefits are:

- Reduces cost of development by eliminating the need to develop a set of services
- Reduces cost of SOA across the organization over time since services do not need to be redeveloped for multiple solutions now and in the future
- increases agility where the service can be used in unanticipated ways
- Decrease development time to value since you are using existing services that do not have to be developed
- Increases agility since the services are available for rapid development
- Decreases risk since the services have been well tested before being reused

4.1.5 Discoverable

The principle of discoverability is achieved when it is possible to find out about the existence, location, and/or description of a service, usually in preparation of a service interaction.

Services may be discovered at design or runtime in the service lifecycle.

The capability to locate or discover services based on specific criteria is enabled by registries and repositories.

Service consumers must fundamentally be able to find services and be able to interact before the rest of management, security and governance really matters.

The most basic service discovery infrastructure for a service domain allows *direct* interaction between service requesters and service implementations. For direct interconnections, the service information could be provided from a-priori knowledge directly into the implementation. This is tightly coupled. Usually, even this simple static connection uses a service registry into which metadata advertising the services available to consumers has been *published*. In this static case, the registry provides this information only in early stages of the service lifecycle, i.e., allowing services to be discovered and selected during model, assemble and deploy. However, service selection during runtime by looking up the endpoints dynamically from registry (*late binding*) increases loose coupling.

Using late binding via a registry also allows the service consumer to be isolated from service location changes. Combining use of policy with late binding further isolates the consumer from the service implementation by allowing configuration of the service interaction. Finally, adding the use of an integration layer isolates both the consumer, service implementation and increases reliability and availability. This combination of features enables changes in capacity, scaling, and quality of service as the needs of the business demand without impacting the consumer or service implementations.

Facilitated by:

- Storing service description artifacts in a repository and/or registry
- Providing search facilities/tools to be able search for, find and access the service description
- Governance processes dictating registration and storage of service descriptions

Benefits are:

- Enables late binding to services, which increases robustness
- Enables loose coupling by providing location and access to the service description
- Increases robustness and agility of the SOA solution by enabling late binding
- Enables reuse of services by ensuring that existing, reusable services and service descriptions can be located and used

4.1.6 Composable

The principle of composability is achieved when services can be an element of a composition without having to change the implementation of the service.

Composability is independent from the kinds of compositions that can be created, processes, choreographies, orchestrations, etc.

Facilitated by:

- Adhering to the principle of reuse
- Developing autonomous services

Benefits are:

- Agility is promoted by reusing services to develop new service
- Enables the development of compositions, processes, choreographies, and orchestrations

4.1.7 Self-Contained

The principle of self-containment is achieved when a service can be invoked with only the information available in the services description. The service consumer should be isolated from the implementation details of the service. Self-Contained services are encapsulated and do not depend on other services for their state or are stateless.

Facilitated by:

- Isolating the consumer from the implementation (opacity)
- Developing self-contained services
- Developing complete descriptions
- Providing an implementation independent interface

Benefits are:

- Increases reusability
- Increased composability

4.1.8 Loosely coupled

Loose Coupling is achieved when service consumption is insulated from underlying implementation.

This clause defines a set of capabilities needed to manage and increase loose coupling in SOA solutions.

In most industries anything that is reusable allows some variability and flexibility when connecting to other elements, such as expansion joints in the building industry, backlashes in the mechanical industry, and so on. In the information technology industry, this allowance for variability and flexibility are referred to as loose coupling.

The service-oriented architecture (SOA) architectural style supports loosely coupling the interface of a service being consumed from the implementation being provided.

Loose Coupling facilitated a number of ways:

- By requiring consumers to use interfaces and/or contracts for interacting with services ensures that the service features can be fulfilled by any available implementation rather than a particular instance of an implementation
- By separating implementation from both its binding and service endpoint interface
- By having service consumers use late binding (at runtime) to service instances via service discovery, which can be provided by a registry, repository, or mediation layer
- By externalizing configuration and runtime contracts into policy
- Using an integration layer to provide support for connections, protocol mediation, security and other qualities of service. This relieves the consumer and service implementations from having to provide this directly or having to implement quality of service 'matching' directly.

The benefits of loose coupling are:

- Minimizing changes to consumers of services over time, even when versions change or changes are needed for qualities of service or protocol support.
- Minimizing changes to services to change versions, protocols, capacity, and qualities of service
- Minimizing changes to service implementations to support reuse of services in new opportunities, increasing business agility
- Increasing service availability and reliability for consumers by allowing selection of appropriate service with late binding at runtime
- Decreasing cost by enabling reuse of services
- Enabling the addition and benefits of a integration layer
- Enabling federation of SOA domains – sharing of services from one SOA solution domain to another
- Enabling management of services to be policy driven
- Enabling reuse of services by isolating configuration into policy

To achieve these benefits of loose coupling, some additional management of service metadata, will be needed.

4.1.9 Manageable

The principle of manageability is achieved when services and solutions can be developed, controlled, monitored, configured, and governed such that policies, contracts and agreements are adhered to.

Facilitated by:

- Defining and monitoring key performance metrics for availability, reliability, performance and governance
- Providing interfaces for management capabilities along with interfaces for functional capabilities
- Enabling management of services to be policy driven
- Isolating configuration into policy
- Enabling configuration of services at design and runtime

- Enabling monitoring of adhering to contracts
- Development of governance policies and processes
- Enabling control over the lifecycle of the service, (enable, disable)

Benefits are:

- Awareness of service availability
- Reduced risk of service failure
- Contract enforcement
- Governance process execution and automation
- Alignment of business and IT requirements for the SOA solution
- Automation of service and SOA solution management

4.2 Meta Model

The SOA Reference Architecture has nine layers representing nine 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 discussed in clause 3.

Taking a pragmatic approach, for each layer, there are three aspects that should be supported by the SOA RA: requirements (exemplified by the capabilities for each layer), logical (exemplified by the architectural building blocks) and physical (this aspect will be left to the implementation of the standard by an adaptor of the standard).

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

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

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

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

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

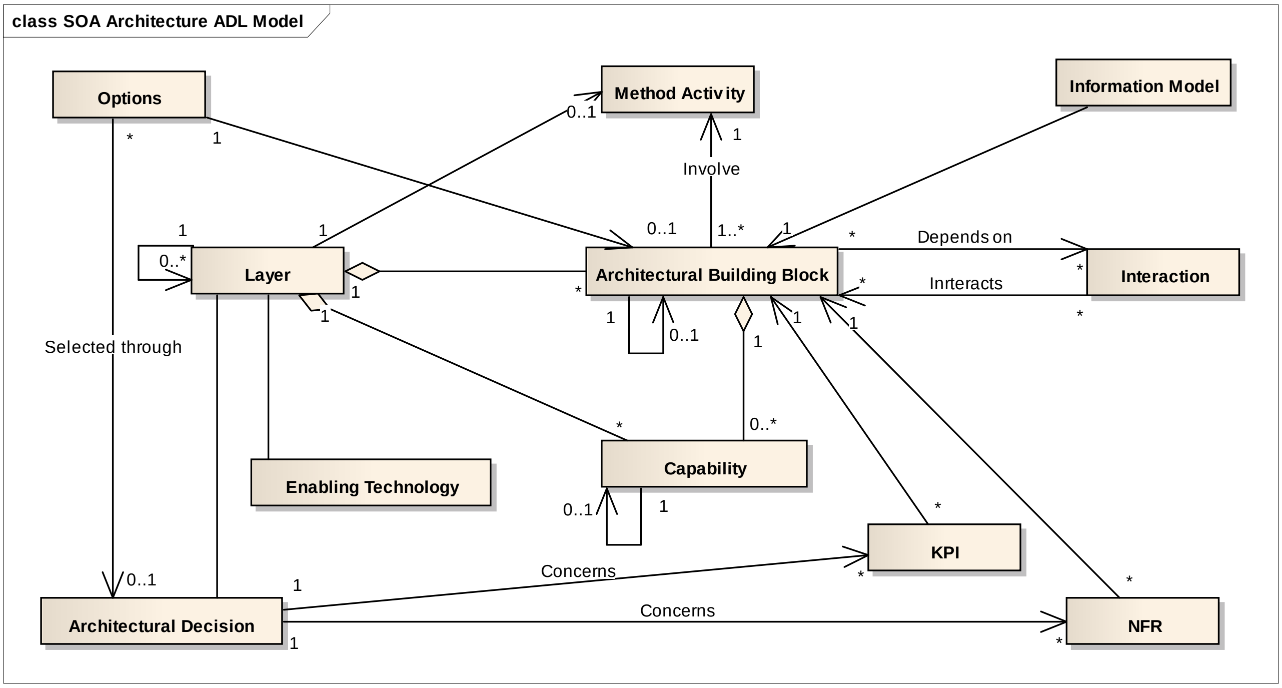

Figure 11 Meta-model for Instantiating the SOA Reference Architecture for a Given Solution

As shown in Figure 11, the meta-model of the SOA Reference Architecture includes the following elements:

- *Layer*: An abstraction which contains a set of components such as architectural building blocks (ABB), architectural decisions, interactions among components and interactions among layers, and supports a set of capabilities.
- *Capability*: An ability that an organization, person, or system possesses to deliver a product or service. A capability represents a requirement or category of requirements that fulfill a strongly cohesive set of needs. This cohesive set of needs or functionality is summarized by name given to the capability.
- *ABB (Architectural building block)*: A constituent of the architecture model that describes a single aspect of the overall model [12]. Each layer can be thought to contain a set of ABBs that define the key responsibilities of that layer. In addition, ABBs are connected to one another across layers and thus provide a natural definition of the association between layers. The particular connection between architectural building blocks that recur consistently in order to solve certain classes of problems can be thought of as patterns of architectural building blocks. These patterns will consist not only on a static configuration which represents the cardinality of the relationship between building blocks, but also the valid interaction sequences between the architectural building blocks. In this reference architecture each ABB resides in a layer, supports capabilities, and has responsibilities. It contains attributes, dependencies, constraints and relationships with other ABBs in the same layer or different layer.
- *Method activity*: A set of steps that involve the definition or design associated with ABBs within a given layer. The method activity provides a dynamic view of how different ABBs within a layer will interact. Method activities can also be used to describe the interaction between layers, so that an entire interaction from a service invocation to service consumption is addressed.
- *Options*: A collection of possible choices available in each layer that impact other artifacts of a layer. Options are the basis for architectural decisions within and between layers, and have concrete standards, protocols, and potentially solutions associated with them. An example of an option would be choosing SOAP or REST style SOA Services since they are both viable options. The selected option leads to an architectural decision.

- *Architectural decision*: A decision derived from the options. The architectural decision is driven by architectural requirements, and involves governance rules and standards, ABBs, KPIs (Key Performance Indicators) and NFRs (Non Functional Requirements) to decide on standards and protocols to define a particular instance of an Architectural Building Block. This can be extended, based on the instantiation of the Reference Architecture to the configuration and usage of ABBs. Existing architectural decisions can also be reused by other layers or ABBs.
- *Interaction pattern*: An abstraction of the various relationships between ABBs. This includes diagrams, patterns, pattern languages and interaction protocols.
- *KPI (Key performance indicator*): A key performance indicator may act as input to an architectural decision.
- *NFR (Non-functional requirement)*: An NFR may act as input to an architectural decision. NFRs help address Service Level Agreement attributes, (e.g. response time, etc.) and architectural cross-cutting concerns such as security.
- *Enabling Technology*: A technical realization of ABBs in a specific layer.
- *Information Model:* A structural model of the information associated with ABBs including data exchange between layers and external services. The information model includes the meta-data about the data being exchanged.

4.3 Capabilities

A capability, as defined by the Open Group, is "*an ability that an organization, person, or system possesses*" [TOGAF 9 XX]. From a TOGAF context "capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve. Marketing, customer contact, outbound telemarketing etc. are illustrative examples for capabilities. The term capability can represent pure business capability such as Process Claim or Provision Service Request and can also represent technical capability such as Service Mediation or Content Based Routing. Both business and technical capabilities are represented in SOA and enabled and supported by SOA. Using a capability modelling as part of the approach has some major advantages:

1. Allows us to focus on the "what" rather than the "how". This supports an abstract approach that is focused on the requirements of the solution.

2. Enables business capabilities to be aligned to the technical capabilities required to service them. Through the use of The Open Group SOA RA the associated enterprise level and solution architectures can then be derived.

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

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

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

A capability based approach enables us to answer the question “When do we need a particular SOA RA layer?” and to help facilitate making decisions when organizational priorities change. The SOA RA further helps us by determining if there are interdependencies and technical requirements for a layer and its constituent building blocks, beyond those defined by business capabilities, to create a holistic set of capabilities which the SOA RA needs to satisfy. Figure 12 below show the relationships among the core constructs of the SOA RA.

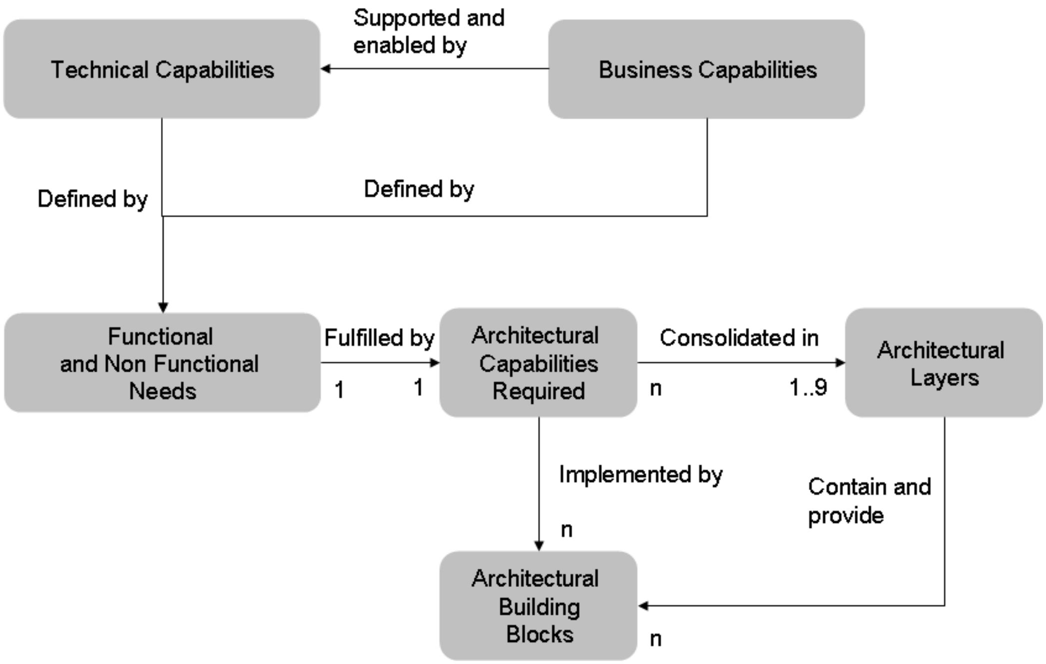

Figure 12 Relationships among Requirements, Capabilities, Building Blocks and Layers

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

4.4 Assumptions

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

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

- A set of service requirements which includes business (*aka* 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 fulfillment of any service requirement may be achieved through the capabilities of a combination of one or more layers in the SOA Reference Architecture (SOA RA).

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

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

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

- Introduction: Provide an overview of the layer itself.

- Requirements: Provide an understanding of the capabilities supported by the layer and what they are (the answer to the "what does the layer do" question).

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

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

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

- Provide an overview and description of the layer and motivation behind the layer

- Provide the key capabilities supported by the layer

- Provide a structural overview of the layer which includes detailed description of ABBs enabling the responsibilities of the layer

- Describe the interactions within the layer and across other layers in the SOA RA

Figure 13 describes an interaction between the Consumer Layer and the Business Process Layer using the Integration Layer.

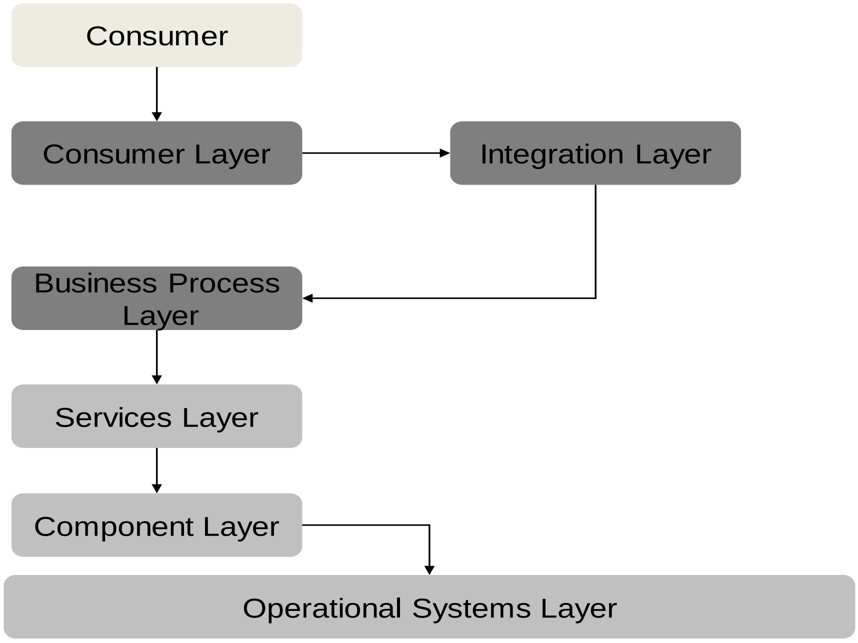

Figure 13: Typical Interactions among the Layers of the SOA RA

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

- Service consumers request services using the Integration Layer.
- The Integration Layer invokes the business process in the Business Process Layer which is using one or more services.
- It 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 Systems Layer to carry out the service request.
- The response is sent back up to the service consumer.

Annex A
(Informative)

Bibliography

1. ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*, 2001

2. ISO/IEC TR 10000-1, *Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework*

3. ISO 10241, *International terminology standards — Preparation and layout*

4. ISO 128-30, *Technical drawings — General principles of presentation — Part 30: Basic conventions for views*

5. ISO 128-34, *Technical drawings — General principles of presentation — Part 34: Views on mechanical engineering drawings*

6. ISO 128-40, *Technical drawings — General principles of presentation — Part 40: Basic conventions for cuts and sections*

7. ISO 128-44, *Technical drawings — General principles of presentation — Part 44: Sections on mechanical engineering drawings*

8. ISO 31 (all parts), *Quantities and units*

9. IEC 60027 (all parts), *Letter symbols to be used in electrical technology*

4. ISO 1000, *SI units and recommendations for the use of their multiples and of certain other units*

5. ISO 690, *Documentation — Bibliographic references — Content, form and structure*

6. ISO 690-2*, Information and documentation — Bibliographic references — Part 2: Electronic documents or parts thereof*

7. ISO/IEC 15288, *System life cycle processes*

8. ISO/IEC 12207, *Software life cycle processes*

9. ISO/IEC 42010, *Architecture description*

10. ISO/IEC 10746, *Reference Model For Open Distributed Processing*

11. ISO/IEC JTC 1/SC 38 N0043, *Research Report on China's SOA Standards System*

12. ISO/IEC JTC 1/SC 38 N0022, *Chinese National Body Contribution on Proposed NP for General Technical Requirement of Service Oriented Architecture*

13. 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

14. 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

15. The Open Group, Technical Standard Service-Oriented Architecture *Ontology* Available from World Wide Web: http://www.opengroup.org/soa/source-book/ontology/index.htm,

16. Open Group Technical Standard, *SOA Governance Framework,* Available from World Wide Web: http://www.opengroup.org/soa/source-book/gov/intro.htm,

17. OMG *Business Process Management Notation* (BPMN), see http://www.omg.org/spec/BPMN/2.0/

18. ISO Technical Report TR9007, *Concepts and Terminology for the Conceptual Schema and the Information Base*

19. *The Open Group Architecture Framework (TOGAF*), section 8.1.1 Version 9 Enterprise Edition, February 2009; see www.opengroup.org/togaf

20. OASIS *Reference Architecture for SOA Foundation,* Version 1.0, OASIS Public Review Draft 1, April 2008: see docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf

21. W3C *Web Services Description Language* (WSDL) 1.1, W3C Note 15 March 2001, see http://www.w3.org/TR/wsdl

22. 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

23. OMG *Model Driven Architecture (MDA) Guide,* Version 1.0.1, Object Management Group (OMG), June 2003: see www.omg.org/docs/omg/03-06-01.pdf

24. 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

25. OMG *SOA Modeling Language* (OMG SoaML) Specification for the UML Profile and Metamodel for Services (UPMS), Revised Submission, OMG Doc. No.: ad/2008-11-01, Object Management Group (OMG), November 2008: see www.omg.org/cgi-bin/doc?ad/08-11-01

26. W3C *Web Ontology Language (OWL),* World Wide Web Consortium (W3C), April 2009: see www.w3.org/2007/OWL/wiki/OWL_Working_Group

27. Garrett, Jesse James, *A New Approach to Web Applications,* Feb 18, 2005 see http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

28. 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

29. 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

30. 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

Annex B

(Informative)

Issues List

The following issues remain to be addressed:

Comment Ref	Comment summary	Action/Disposition
CA04	Concepts and definition should be introduced in Clause 4 from primitive (atomic) concepts to more complex in clause 4. This would facilitate review and understanding. Furthermore they are not structured in an explicit way, and some of them are outside the scope of SC38. This was raised in PDTR comment CA-065 and other. As recommended, new text is proposed. The atomic concepts used in the definition of SOA were not defined. A new section 4.1 is proposed to that effect. Note that this proposal requires additional definitions in clause 3.2. See CA-005 please refer to attachment CA-004 later in this document for text that: 1. introduces basic concepts 2. proposes a classification for concepts 3. discusses scope	WG2 invites national body comments on this comment
CA05	The following definitions are required for the new proposed text for 4.1 please refer to attachment CA-005 later in this document for the proposed text	WG2 invites national body comments on this comment
CA06	The concepts in section 4 were not introduced according to the system (business, application, technology) they applied to. This distinction is important since some of the material is out of scope for SC38. Rearranged text is proposed for the remainder of clause 4 Please refer to attachment CA-006 later in this document for the proposed text. Note that original numbering has been kept to facilitate the editor's work	WG2 invites national body comments on this comment

CA08	The content of clause 5 does not qualify as an architecture description as per the requirements stated in clause 5 of 42010. The content of clause 5 does not follow any recognizable architecture framework (for instance TOGAF, or OMG MDA, or ISO ODP), nor is the actual architectural framework and modelling conventions introduced. The content of clause 5 does not qualify as"reference" architecture. RAs have to be abstract, stable, follow explicit modelling rules so that they can be proven correct, Clause 5 is not required to fulfil the original main objective and title of the project and the document, that is "General Technical Principles" Remove Clause 5. Restructure the document so that Clause 4 should be Concepts and Clause 5 Principles. Adjust introductory text accordingly.	WG2 invites national body comments on this comment
CA07	The following material deals with the business (and IT business aspects) and is deemed out of scope for SC38 and clause 4, and is put in an annex so that the material is not lost at this stage. Is felt that this material is more suitable for the "cloud" reference architecture (SC38 WD 17789) please refer to attachment CA-007 later in this document for the proposed text	WG2 invites national body comments on this comment

