INTERNATIONAL STANDARD

ISO/IEC 18384-1

First edition
2016-06-01

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

Part 1: **Terminology and concepts for SOA**

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

Partie 1: Terminologie et concepts pour SOA

Reference number
ISO/IEC 18384-1:2016(E)

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

Contents

Page

Foreword

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

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

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

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

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

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

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

— *Part 1: Terminology and Concepts for SOA*

— *Part 2: Reference Architecture for SOA Solutions*

— *Part 3: Service Oriented Architecture Ontology*

Introduction

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

SOA takes "service" as its basic element to constitute and integrate information systems so that they are suitable for a variety of solution requirements. SOA enables interactions between businesses without needing to specify aspects of any particular business domain. Using the SOA architectural style can improve the efficiency of developing information systems, and integrating and reusing IT resources. In addition, using the SOA architectural style can help realize agile and rapid response of information systems to ever-changing business needs.

This International Standard describes a single set of SOA technical principles, specific norms, and standards for the world-wide market to help remove confusion about SOA and improve the standardization and quality of solutions.

This International Standard defines the terminology, technical principles, reference architecture, and the ontology for SOA. The targeted audience of this International Standard includes, but is not limited to, standards organizations, architects, architecture methodologists, system and software designers, business people, SOA service providers, SOA solution and service developers, and SOA service consumers who are interested in adopting and developing SOA. For example, this part of ISO/IEC 18384 can be used to introduce SOA concepts and to guide to the developing and managing SOA solutions.

This International Standard contains three parts:

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

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

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

Users of this part of ISO/IEC 18384 will find it useful to read this part of ISO/IEC 18384 for an understanding of SOA basics. This part of ISO/IEC 18384 should be read before reading or applying ISO/IEC 18384-2. For those new to SOA, ISO/IEC 18384-2:2016, Clause 4 provides a high level understanding of the reference architecture for SOA solutions. The remaining clauses provide comprehensive details of the architectural building blocks and trade-offs needed for a SOA solution. ISO/IEC 18384-3 contains the SOA Ontology, which is a formalism of the core concepts and terminology of SOA, with mappings to both UML and OWL. The SOA Ontology can be used independent of or in conjunction with ISO/IEC 18384-1 and ISO/IEC 18384-2.

This part of ISO/IEC 18384 presents and explains basic SOA concepts. It gives definitions for terms that are used in ISO/IEC 18384 with specific meanings that may differ or be more precise than the definitions of those terms found in major English language dictionaries. The terms defined here are used in a unique fashion for SOA. Terms used in their normal English sense are not redefined.

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

Part 1:
Terminology and concepts for SOA

1 Scope

This part of ISO/IEC 18384 establishes vocabulary, guidelines, and general technical principles underlying service oriented architecture (SOA), including principles relating to functional design, performance, development, deployment, and management.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1
actor
person or system component that interacts with the system as a whole and that provides stimulus which invokes actions

[SOURCE: ISO/IEC 16500-8:1999, 3.1]

2.2
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

[SOURCE: ISO/IEC/IEEE 42010:2011, 3.2]

2.3
choreography
type of *composition* (2.5) whose *elements* (2.8) interact in a non-directed fashion with each autonomous part knowing and following an observable predefined pattern of behaviour for the entire (global) composition

Note 1 to entry: Choreography does not require complete or perfect knowledge of the pattern of behaviour.

Note 2 to entry: See ISO/IEC 18384-3:2016, 8.3.

2.4
collaboration
type of *composition* (2.5) whose *elements* (2.8) interact in a non-directed fashion, each according to their own plans and purposes without a predefined pattern of behaviour

Note 1 to entry: See ISO/IEC 18384-3:2016, 8.3.

2.5
composition
result of assembling a collection of *elements* (2.8) for a particular purpose

Note 1 to entry: See ISO/IEC 18384-3:2016, 8.2.

2.6
endpoint
location at which information is received to invoke and configure interaction

2.7
effect
outcome of an interaction with a *service* (2.20)

Note 1 to entry: The effect is how a service delivers results to its consumer, through the *element* (2.8) that performs it.

Note 2 to entry: See ISO/IEC 18384-3:2016, 7.10.

2.8
element
unit at a given level of abstraction and with a clearly defined boundary

Note 1 to entry: An element can be any type of *entity* (2.9).

Note 2 to entry: See ISO/IEC 18384-3:2016, 5.1.

2.9
entity
individual *element* (2.8) in a system with an identity which can act as a *service provider* (2.50) or *service consumer* (2.29)

Note 1 to entry: Examples of entities are organizations, enterprises and individuals, software, and hardware.

2.10
event
something that occurs to which an *element* (2.8) may choose to respond

Note 1 to entry: Any element can generate (emit) or respond to an event.

Note 2 to entry: See ISO/IEC 18384-3:2016, Clause 10.

2.11
execution context
set of technical and business *elements* (2.8) needed by those with needs and capabilities to permit *service providers* (2.50) and *service consumers* (2.29) instantiation and communication

Note 1 to entry: The execution context of a *service interaction* (2.37) 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 2 to entry: See Reference [8].

2.12
human actor
actor (2.1) restricted to a person or an organizational *entity* (2.9)

Note 1 to entry: This classification is not exhaustive.

Note 2 to entry: See ISO/IEC 18384-3:2016, 6.2.

2.13
human task
task (2.60) which is done by a *human actor* (2.12)

2.14
interface
shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical interconnections, signal exchanges, and other characteristics, as appropriate

[SOURCE: ISO/IEC 2382:2015, 2121308]

2.15
loose coupling
principle where dependencies between *elements* (2.8) of a *SOA solution* (2.56) are intentionally reduced

2.16
orchestration
type of *composition* (2.5) where one particular *element* (2.8) is used by the composition to oversee and direct the other elements

Note 1 to entry: The element that directs an orchestration is not part of the orchestration (Composition instance) itself.

Note 2 to entry: See ISO/IEC 18384-3:2016, 8.3.

2.17
policy
statement that an *entity* (2.9) intends to follow or intends that another entity should follow

Note 1 to entry: See ISO/IEC 18384-3:2016, Clause 9).

2.18
process
type of *composition* (2.5) whose *elements* (2.8) are composed into a sequence or flow of activities and interactions with the objective of carrying out certain work

Note 1 to entry: A process may also be a *collaboration* (2.4), *choreography* (2.3), or *orchestration* (2.16).

Note 2 to entry: See ISO/IEC 18384-3:2016, 8.6.

2.19
real-world effect
change relevant to and experienced by specific stakeholders

Note 1 to entry: See Reference [8].

2.20
service
logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a "black box" to consumers of the service

Note 1 to entry: The word "activity" in the "service" definition 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* (2.18) activities].

Note 2 to entry: See ISO/IEC 18384-3:2016, 7.2.

2.21
service broker
element (2.8) that enables the communication with *services* (2.20), either at a business level or at the implementation level, i.e. with intermediaries

Note 1 to entry: The intermediaries provide any number of functions, such as unified *service registration* (2.51) and publishing, *service discovery* (2.34), routing, location-transparent service access, for *service providers* (2.50) and *service consumers* (2.29).

2.22
service bus
design and runtime pattern for enabling *service* (2.20) interactions, such as communication, access, consumption, transformation, intermediaries, and message routing

Note 1 to entry: A service bus can range from a logical collection of such functions to the functions collected into a single commercial product. Service bus is widely used in an organizational context and often equates to the enterprise service bus (ESB).

2.23
service candidate
service (2.20) identified during the *SOA lifecycle* (2.58) that meets broad service requirements, and from which one or more are selected for further development as part of an overall *SOA solution* (2.56)

2.24
service catalogue
service registry/repository (reg/rep)
logical collection of *service descriptions* (2.31) and related artefacts that supports publication, registration, search, management, and retrieval of those artefacts

2.25
service choreography
choreography (2.3) whose *elements* (2.8) are *services* (2.20)

Note 1 to entry: See ISO/IEC 18384-3:2016, Clause 8.

2.26
service collaboration
collaboration (2.4) whose *elements* (2.8) are *services* (2.20)

Note 1 to entry: See ISO/IEC 18384-3:2016, Clause 8.

2.27
service component
element (2.8) that implements *services* (2.20)

2.28
service composition
service assembly
composition (2.5) that provides (in the operational sense) higher level *services* (2.20) that are only an assembly of other *services* (2.20)

Note 1 to entry: A composition can support different composition patterns, such as *collaboration* (2.4), *choreography* (2.3), *orchestration* (2.16).

Note 2 to entry: See ISO/IEC 18384-3:2016, Clause 8).

2.29
service consumer
entity (2.9) that uses *services* (2.20)

Note 1 to entry: Consumers may interact with services operationally or contractually (legal responsibility).

Note 2 to entry: See ISO/IEC 18384-3:2016, 7.4.

2.30
service contract
terms, conditions, and interaction rules that interacting *service consumers* (2.29) and *service providers* (2.50) agree to (directly or indirectly)

Note 1 to entry: A service contract is binding on all participants in the interaction, including the *service* (2.20) itself and the *element* (2.8) that provides it for the particular interaction in question.

Note 2 to entry: See ISO/IEC 18384-3:2016, 7.6.

2.31
service description
information needed in order to use, or consider using, a *service* (2.20)

Note 1 to entry: The service description usually includes the *service interfaces* (2.38), contracts, and policies.

Note 2 to entry: See ISO/IEC 18384-3:2016, Clause 7.

2.32
service deployment
activities by which implementations of *services* (2.20) are made able to run in a specific hardware and software environment and usable by *service consumers* (2.29)

2.33
service development
activities by which needs and constraints are identified and *services* (2.20) are designed as part of a *SOA solution* (2.56) in order to address those needs within the constraints

2.34
service discovery
activities by which a *service consumer* (2.29) can find *services* (2.20) which meet their specific functional and/or non-functional requirements

2.35
service governance
strategy and control mechanism that applies across the *service lifecycle* (2.41) and service portfolio, which includes the establishment of chains of responsibility, driving monitoring of compliance with policies by providing appropriate *processes* (2.18) and measurements as part of *SOA solution governance* (2.57)

Note 1 to entry: Aspects of the service lifecycle that need to be governed include: addressing service modifications, version updates, notice of termination, decomposition subdivision, agency capacity, decomposition capacity, and ability to meet individual demands.

2.36
service implementation
activities performing technical development and the physical implementation of the *service* (2.20) that is part of a *service lifecycle* (2.41), and results in the creation of a *service component* (2.27)

2.37
service interaction
activity involved in making use of a capability offered, usually across an ownership boundary, in order to achieve a particular desired *real-world effect* (2.19)

Note 1 to entry: See Reference [8].

2.38
service interface
interface (2.14) by which other *elements* (2.8) can interact and exchange information with the service where the form of the request and the outcome of the request is in the *service description* (2.31)

Note 1 to entry: See ISO/IEC 18384-3:2016, 7.13.

2.39
service interoperability
ability of *service providers* (2.50) and *service consumers* (2.29) to communicate, invoke *services* (2.20) and exchange information at both the syntactic and semantic level leading to effects as defined by the *service description* (2.31)

2.40
service level agreement
SLA
type of *service contract* (2.30) that defines measureable conditions of interactions between a *service provider* (2.50) and a *service consumer* (2.29)

Note 1 to entry: A service level agreement may specify: the set of *services* (2.20) the service provider will deliver, a sufficient, specific definition of each service, the responsibilities of the service provider and the service consumer, the set of metrics to determine whether the service provider is delivering the service as promised, an auditing mechanism to monitor the service, the remedies available to the service consumer and service provider if the terms of the SLA are not met, and how the SLA will change over time.

2.41
service lifecycle
set of phases for realizing a *service* (2.20) that can go through from conception and identification to instantiation and retirement

2.42
service management
monitoring, controlling, maintaining, optimizing, and operating *services* (2.20)

2.43
service modeling
set of activities to develop a series of *service candidates* (2.23) for functions or actions on a *SOA solution* (2.56) using *service oriented analysis* (2.47) processes

2.44
service monitoring
tracking state and operational conditions related to the execution, performance, and *real-world effects* (2.19) of *services* (2.20)

2.45
service orchestration
orchestration (2.16) where the orchestrated *elements* (2.8) are *services* (2.20)

2.46
service orientation
approach to designing systems in terms of *services* (2.20) and service-based development

2.47
service oriented analysis
preparatory information gathering steps that are completed in support of a *service modeling* (2.43) sub-process that results in the creation of a set of *services* (2.20)

Note 1 to entry: It provides guidance to the subsequent phases of the SOA lifecycle and might be carried out just once for each business *process* (2.18) or iteratively.

2.48
service oriented architecture
SOA
architectural style that supports *service orientation* (2.46) and is a paradigm for building business solutions

Note 1 to entry: *Services* (2.20) realized in this style utilize activities that comprise business *processes* (2.18), have descriptions to provide context, may be implemented via *service composition* (2.28), have environment-specific implementations which are described in the context that constrains or enables them, require governance, and place requirements on the infrastructure to achieve interoperability and location transparency using standards to the greatest extent possible.

Note 2 to entry: See ISO/IEC 18384-3:2016, Clause 4.

2.49
service policy
policy (2.17) as applied to a *service* (2.20)

2.50
service provider
entity (2.9) providing *services* (2.20)

Note 1 to entry: Service providers may be responsible for the operation of the services or the contract for the services (legal responsibility) or both.

Note 2 to entry: See ISO/IEC 18384-3:2016, 7.4.

2.51
service publishing
service registration
cataloguing of *service descriptions* (2.31) in an accessible location, such as a *service registry/repository* (2.24), where supporting activities, such as search and retrieval of descriptions, make service information visible and available to potential *service consumers* (2.29)

2.52
SOA implementation
methods and techniques used to develop *SOA* (2.48) based solutions

2.53
SOA maturity
assessment of an organization's ability to adopt *SOA* (2.48) and the current level of adoption

2.54
SOA maturity model
framework stating overall objectives and a method to evaluate an organization's *SOA maturity* (2.53) against these objectives

2.55
SOA resource
elements (2.8) that provide the IT resources used by *services* (2.20)

2.56
SOA solution
solutions, in part or as a whole, implemented by applying *SOA* (2.48) principles, concepts, methods, and techniques

Note 1 to entry: The SOA solutions include the physical instantiation of *service implementations* (2.36), including infrastructure, other architectural elements, and capabilities needed to support governance and lifecycle processes, that together enable domain-specific effects that represent a SOA-based solution to business problems.

2.57
SOA solution governance
specialization of IT governance specifically focused on management strategies and mechanisms for the end users' specific *SOA solution* (2.56)

Note 1 to entry: SOA solution governance manages the entire *SOA solution lifecycle* (2.58) by setting out personnel, roles, management procedures, and decision-making. SOA solution governance needs to adopt the appropriate methodology and best practices. SOA solution governance usually requires tools for assistance to customize and manage the governance strategy according to the needs.

Note 2 to entry: While management means the specific *process* (2.18) 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.

2.58
SOA solution lifecycle
set of activities for engineering *SOA solutions* (2.56), including analysis, design, implementation, deployment, test, and management

2.59
SOA solution management
measurement, monitoring, and configuration of the entire lifecycle of a *SOA solution* (2.56)

Note 1 to entry: At runtime, it is the set of activities for the specific measurement and operation of the implementation of the SOA solution according to the strategies and mechanisms identified by the *SOA solution governance* (2.57) process.

2.60
task
atomic action which accomplishes a defined result

Note 1 to entry: Tasks are done by people or organizations, specifically by *human actors* (2.12).

Note 2 to entry: See ISO/IEC 18384-3:2016, 6.4.

2.61
web services
software system designed to support interoperable machine-to-machine interaction over a network

Note 1 to entry: Original definition: Software system designed to support interoperable machine-to-machine interaction over a network. It has an *interface* (2.14) 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 2 to entry: See Reference [18].

3 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

ABB	Architectural Building Block
BPMN	Business Process Model and Notation
COBIT	Control Objectives for Information and Related Technology
EA	Enterprise Architecture
ESB	Enterprise Service Bus
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IT	Information Technology
ITIL	Information Technology Infrastructure Library
KPI	Key Performance Indicator
L2TP	Layer 2 Tunneling Protocol
MPLS	Multiprotocol Label Switching
OWL	Web Ontology Language
PKI	Public Key Infrastructure
QoS	Quality of Service
RA	Reference Architecture
REST	Representational State Transfer
RPC	Remote Procedure Call

SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Standard message protocol to exchange structured data
SQL	Structured Query Language
UML	Unified Modeling Language
VPN	Virtual Private Network
WSDL	Web Services Description Language
WSRP	Web Services Remote Portlet
XML	Extensible Markup Language

4 Notations

4.1 General

The following provides instruction on the interpretation of diagrams.

4.2 UML

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

4.3 Entity Relationship

Entity relationship diagrams (like Figure 1 and Figure 2) with boxes, lines, arrows, and circled numbers should be interpreted according to the following rules.

- Boxes are the metamodel concepts, layers, architectural building blocks, capabilities, or components.
- Arrows are relationships between metamodel concepts; single arrow heads show direction of relationship; double headed arrows indicate the relationship is bidirectional.
- Relationships are named, represented as labelled lines or arrows, and no cardinality is implied.
- Cardinality indications are participation in the relationship, well-known mathematical conventions are used (* = = 0..*; 0..1 = = optional and only 1; 1 = = required as defined in ISO/IEC 15474-1).

4.4 Cycles

Circular diagrams with states in them show the progression of a state or lifecycle and progress clockwise. Figure 10 is an example of a cycle diagram.

4.5 Flows

Flows are often used for examples and should be interpreted with the following rules:

- boxes that are layers, architectural building blocks, or components;
- directional arrows showing the direction of the flow between the boxes;
- circled numbers on the flow arrows show the sequence of the flow and are used as a point of reference in any explanatory text.

Figure 5 and Figure 6 are flows used as examples.

5 Conventions

This reference architecture is defined in three parts. This part of ISO/IEC 18384 defines the terminology and concepts for service oriented architecture. Understanding the terms and concepts defined in this part of ISO/IEC 18384 is important before reading or using ISO/IEC 18384-2 and ISO/IEC 18384-3. This part of ISO/IEC 18384 defines a reference architecture for SOA solutions. It defines a metamodel, a set of layers for the layered architecture, and a set of common service types. ISO/IEC 18384-3 defines the ontology for SOA, with a more formal expression of core SOA concepts and relationships between them. The terminology in this part of ISO/IEC 18384 is consistent with the ontology in ISO/IEC 18384-3.

This International Standard can be read in sequence or used as a reference. This part of ISO/IEC 18384 contains vocabulary and thorough introductory material and SOA concepts (this part of ISO/IEC 18384). ISO/IEC 18384-2 provides a short introduction to SOA. Introduction is followed by a high level overview of the 10 layers and the service types defined in this International Standard. This is followed by the definition and explanation of the metamodel used in the SOA RA. The metamodel defines layer, capabilities, and ABB concepts along with other core logical concepts. ABBs and capabilities are each defined uniquely in each layer. Capabilities and ABBs may require capabilities and ABBs defined in other layers in order to do fulfil their architectural requirements. The layers, capabilities, and ABBs in this part of ISO/IEC 18384 are all logical elements and any reference to these logical elements "performing", "supporting", or "interacting" means that when a SOA solution is developed, the physical realization of the capabilities and ABBs are actually "performing", "supporting", or "interacting".

6 Conformance

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

a) terminology and concepts — conformance only to terms and adherence to the semantics in the definitions;

b) reference architecture for SOA solutions — conformance only to semantics of the metamodel and any layers, ABBs, or capabilities that are used;

c) SOA Ontology — conformance for OWL or non-OWL applications.

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

If any document, product, or standard claims conformance, then the terms in this part of ISO/IEC 18384 shall be used with the same semantics as these definitions.

Principles and concepts are provided as explanatory material and not meant for performance.

7 SOA Concepts

7.1 Introduction to SOA

Service oriented architecture (abbreviated SOA) (see 2.48) is an architectural style that supports service orientation (see 2.46) and is a paradigm for business and IT. This architectural style is for designing systems in terms of services available at an interface and the outcomes of services. As noted in 2.20, a service is a logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services and is a "black box" to consumers of the service.

In common with other architectural styles, SOA

— places unique requirements on system infrastructure,

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

— recommends governance of IT and systems and EA,

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

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

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

a) it promotes the use of open standards and interfaces in order to achieve interoperability and location opacity;

b) services and processes are designed explicitly to operate both within or between organizations;

c) it requires clear descriptions of the service offered;

d) services and processes are designed to mirror real-world business activities;

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

f) it requires appropriate governance of service representation and implementation;

g) service composition is used as a means to implement business processes;

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

SOA takes “service” as its basic element to construct information systems so that they are suitable for a variety of solution requirements. Service from a business perspective is the delivery of business outcomes of business processes; service from an IT perspective is the IT implementation of those business processes. The activities in developing a SOA solution 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).

Some of the intended benefits of using SOA are improvement in the efficiency of development of information systems, efficiency of integration, and efficiency of reuse of resources.

While there is interest in SOA as a means of delivering these efficiencies, 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 increase standardization and potentially the quality of solutions, as well as promote effective large-scale adoption of SOA, it is necessary to establish a unified set of terms, principles, and concepts for SOA.

It should be noted that these SOA principles defined here are applicable to software engineering and can also be applicable to systems engineering in order to formalize service-based systems (i.e. complex systems, federation of systems, systems of systems, enterprise architecture).

7.2 Concepts

7.2.1 Roles

7.2.1.1 Overview

In general, a role is defined by a set of activities that serve a common purpose. When an entity is assigned or assumes a role, it performs the activities of the role. The definition of a role includes whether all of the activities are required and when they are performed. Service providers and service consumers are fundamental roles in a service oriented architecture. A SOA ecosystem comprises services that deliver functionality, service consumers who interact with services, and service providers who develop and host the services for the consumer. However, the roles of service providers and service consumers span a range of activities and may be accomplished by different parties responsible for different activities, i.e. in the following discussions, the party that may have operational responsibility may not have contractual responsibility and vice versa. In addition, a party assuming the service provider role in one context may assume the service consumer role in another. In the following, the term *service provider* is used to indicate an entity performing an activity associated with the service provider role; the term *service consumer* is an entity performing an activity associated with the service consumer role.

7.2.1.2 Service Providers

For service providers, activities include developing, testing, and deploying services that satisfy a range of functional and non-functional requirements. These requirements include operating characteristics and contractual obligations, as well as supporting and responding to communications from other entities in the SOA solution or ecosystem. A service provider monitors the service during operations to assess its compliance with requirements and to identify and mitigate unwanted behaviour. A service provider is responsible for defining and enforcing service governance. In addition, a service provider is responsible for registering the service and maintaining its service description.

There are two contexts under which service providers may have responsibilities when providing services: operationally and contractually. Note that the entity that is contractually responsible (a participant in a contract or service level agreement) may not be the same entity that is operationally responsible (i.e. exchanges messages with the service consumer).

- Operationally — the service provider may engage in activities related to the exchange of messages with the service consumer, as well as producing the promised effect of invoking the service. Entities assuming operational responsibility across the service lifecycle may engage in any of the following:
  - developing services: lifecycle processes for modelling and creating implementations of services, of which service implementation is one step (see the service lifecycle concepts in 7.2.8);
  - deploying services: placing service artefacts and otherwise modifying the services environment such that the service can exchange communications leading to realizing service functionality and associated effects;
  - providing services: supporting use of deployed service through communications exchange and necessary SOA ecosystem support to realize service functionality and service effects;
  - publishing services: creating and maintaining service description (including, as appropriate, service categorization) such that it is searchable and retrievable by all service participants, including service providers, service consumers, and other parties;
  - hosting services: providing and supporting use of infrastructure necessary for developing, deploying, and providing services;
  - governing services: prescribing conditions and constraints consistent with satisfying common goals of service participants and the structures and processes needed to define and respond to actions taken towards realizing those goals;
- Contractually — the service provider may engage in activities related to defining business aspects and legal obligations related to the use of a provided service and coordinating use and the associated agreements with the service consumer as captured in the service contract. Entities assuming contractual responsibility across the service lifecycle may engage in any of the following:
  - pricing service: deciding how to price the services, or how/whether to exploit them for other value;
  - defining service agreements: in cooperation with service consumer, deciding what kind of formal agreements, such as service level agreements (SLAs), are required to use the service, including resolving conflicts in preferred service policies;
  - defining service contracts: capturing service agreements and applicable aspects of governance in an unambiguous manner that facilitates enforcement;

  - enforcing service contracts: engaging in processes, likely defined within the service contract, to ensure fulfilment of the contract or prescribed remedial actions;
  - governing business and contracts: setting up business rules, policies and requirements for contracts, and to enable and monitor operational governance of services.

7.2.1.3 Service Consumers

Service consumers engage in activities to identify services that meet their needs, assess whether they can meet prescribed conditions of use (e.g. access permissions, licensing, and magnitude of requests), understand the information exchange and protocols that enable communications with the service, and be able to engage network capabilities to successfully carry out communications. A service consumer should be able to access and understand the service description, both as it relates to originally identifying the service and to complying with changes to specifics of service interaction or conditions of use.

A service provider can also simultaneously act as a service consumer. In the case of a composite service, the service provider responds to an external service consumer but then becomes the service consumer of the component services to realize the documented effects that are to be produced by the composition.

There are two contexts under which service consumers may have responsibilities when consuming services: operationally and contractually. As with service providers, the entity that is contractually responsible (a participant in a contract or service level agreement) may not be the same entity that is operationally responsible (i.e. exchanges messages with the provider). Note that the operational and contractual activities of the service consumer are often direct counterparts of provider activities.

- Operationally — the service consumer may engage in activities related to the discovery of services and the exchange of messages with the service provider. Entities assuming operational responsibility across the service lifecycle may engage in any of the following:
  - discovering services: searching and otherwise examining available service descriptions to identify services that meet their needs (e.g. via interacting with a service registry/repository or consulting consumer recommendations);
  - invoking service: exchange messages with a provided service in order to invoke its functionality and realize its corresponding effects;
  - governing service use: setting and enforcing business polices for the use of and contracting of services so that business needs are met;
- Contractually — the service consumer may engage in activities related to defining business aspects and legal obligations of related to their use of a provided service and coordinating the associated agreements with the service provider as captured in the service contract. Entities assuming contractual responsibility across the service lifecycle may engage in any of the following:
  - contracting: setting and abiding by the contracts;
  - service payment: complying with the service agreements specifying the exchange of funds or services in return for using provided services;
  - defining service agreements: in cooperation with service provider, decide what kind of formal agreements, such as service level agreements (SLAs), are specified as constraints or conditions of use of the service, including resolving conflicts in preferred service policies;
  - defining service contracts: capturing service agreements and applicable aspects of governance in an unambiguous manner that facilitates enforcement;
  - enforcing service contracts: engaging in processes, likely defined within the service contract, to ensure fulfilment of the contract or prescribed remedial actions;
  - governing contracts: ensuring contracts are aligned with evolving business needs and enforced operationally.

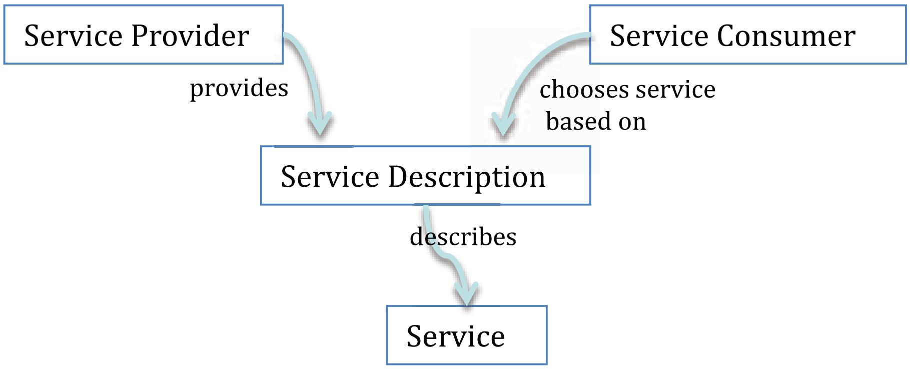

Figure 1 — Service Providers and Service Consumers

7.2.2 Services

As defined in this part of ISO/IEC 18384, a service (see 2.20) is a logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a "black box" to consumers of the service (see ISO/IEC 18384-3:2016, 7.4). Service is agnostic to whether the concept is applied to the business domain or the IT domain. A service can have one or more service providers or service consumers, and produces the specified outcomes.

The service consumer does not know how the service is implemented. As shown in Figure 1, consumers use the description to choose a service from a provider. If two services have the same service description and produce the same effects when given the same inputs, they are functionally equivalent to the service consumer and can be used interchangeably. To a service provider, a service is a means of exposing capabilities and the implementation determines equivalency.

As a service itself is only a logical representation, any service should have something that can execute the processes and communications necessary to realize the resulting effects. This execution is referred to as "performing the service". Services can be performed by any type of element, including software components, human actors, and tasks. The something that performs a service is 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 interacting with a service may, for example, perform the following 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" a service bus (see 2.22)];

— interact with the chosen element that performs the chosen service (often also facilitated by a service bus).

Concepts, such as service mediations, service proxies, services buses, etc. are used to describe and implement the operational aspects of SOA systems. All of these can be captured as an element representing the service. This representation provides a level of indirection that is critical when we do not want to bind operationally to a particular service endpoint, and enables us to preserve loose coupling and the ability to switch embodiments as needed. Understanding what a service represents and is represented by allows encapsulation of the relatively complex operational interaction (picking the service, picking an element that performs the service, and interacting with that chosen element).

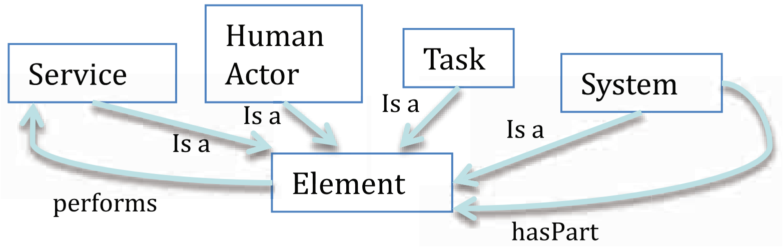

Figure 2 — Service and elements of SOA

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 parts which can be any element, including services, human actors, tasks, and systems themselves. Further explanation of tasks and systems are in the following clauses. (See ISO/IEC 18384-3:2016, Clause 5 for further explanation on systems and SOA systems.)

7.2.3 Semantics

All actors involved in a service oriented architecture solution, human and non-human, should be able to parse messages correctly through a common and agreed upon understanding of the syntax and semantics of the message. Syntax includes interfaces, protocols, and message formats. Semantics is the shared understanding of intent, where actors should be able to act on the contents of those messages in a manner consistent with the sender's intent. In other words, actors should possess a shared semantic understanding of the business requirements being addressed by the service and not just aim for interoperability through a syntactic connection.

In addition, semantics is important in the description of services and other resources in the SOA solution. Discovery requires a commonly understood basis for matching service consumer needs to the real world effects resulting from a service interaction. It is also necessary for unambiguous understanding of conditions of use that should be satisfied for the service interaction to proceed.

Semantics can be formally specified through the use of ontologies and their application to particular usage domains that address a community of concepts.

7.2.4 Tasks and Activities

A task is an atomic action which accomplishes a defined result. Tasks, including Human tasks, are done by people or organizations, specifically by instances of human actor. Because tasks are atomic, tasks cannot be broken down to a finer level of detail and are performed by at most one instance of human actor.

The word "activity" in the "service" definition 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 (see Reference [14]) 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. For SOA, these tasks are not restricted to a process flow.

7.2.5 Compositions and Processes

7.2.5.1 Introduction to Compositions

A composition is a system that is the result of assembling a collection of things for a particular purpose. In this case, composition refers to a collection of parts assembled for a purpose and 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 may be composed of other systems, compositions may be composed of other compositions. No composition can be a part of itself. Since a composition is a collection, it uses at least one other element and those elements can be outside its own boundary. For SOA, these elements are usually, but are not limited to, services, other compositions, processes, actors, and tasks.

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

7.2.5.2 Patterns of Composition

As shown in Figure 3, compositions can be realized using three 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. In Figure 3, the arrows show the flow of direction or invocation between elements in a composition.

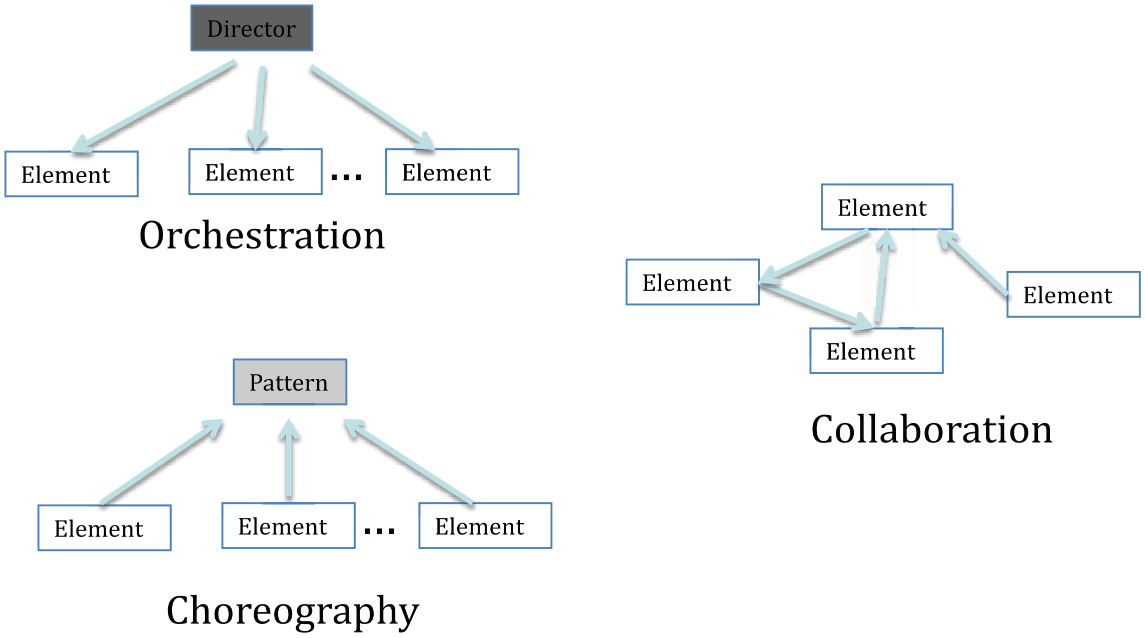

Figure 3 — Patterns of Composition

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 may be inside or outside the boundary. For example, a workflow is a part of the composition but is executing the flow. In Figure 3, the director is invoking elements in the orchestration.

Choreography is a pattern for compositions where the elements used by the composition interact in a non-directed fashion (no director) and every part knowing and following the pattern of behaviour. There is a predefined shared pattern or flow of behaviour. Choreography does not require the elements to have complete knowledge of the pattern of behaviour. In Figure 3, the elements interact with each other according to a pattern.

Collaboration is a pattern for compositions where the elements used by the composition interact in a non-directed fashion (no director), and every part acts according to their own plan/purpose without a predefined pattern or flow of behaviour. Interactions between parts occur as needed by each part. In Figure 3, the elements in the collaboration interact with each other — but there is no pattern or director.

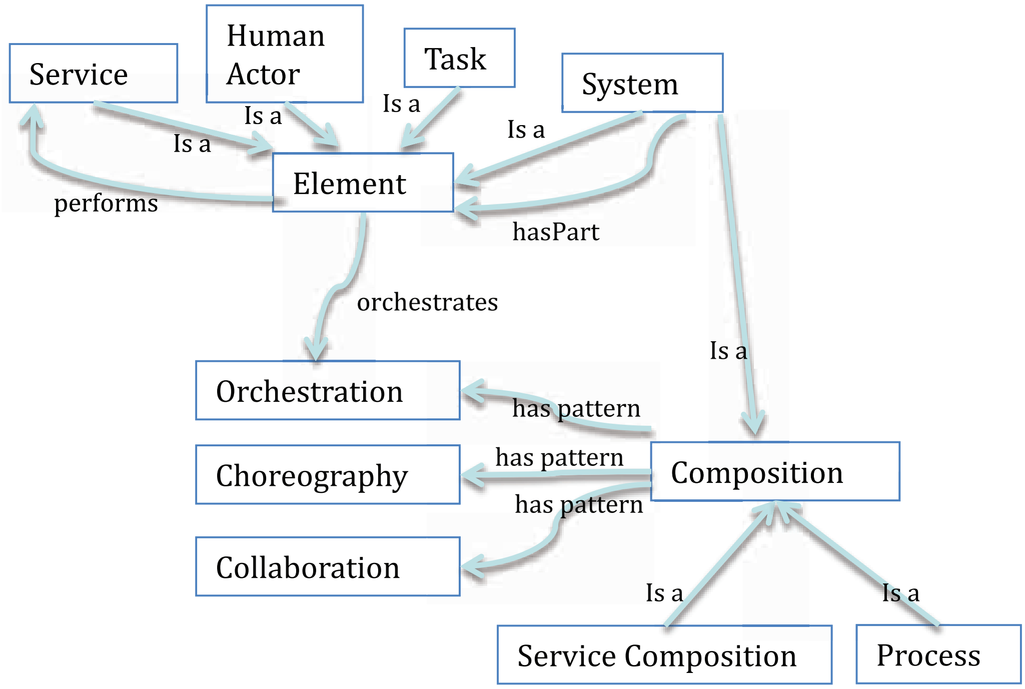

Figure 4 — Composition and its sub-classes

Figure 4 ties together the concepts of service, system, and composition. Elements of systems are services, human actors, tasks, and other systems. Any of these elements can perform a service. A system can have any of these elements as parts of the system. Compositions are systems and therefore can have parts that are services, actors, tasks, and systems. Compositions have a property that indicates the composition pattern it supports: orchestration, choreography, or collaboration. Only compositions which use the orchestration pattern have an element that orchestrates the parts of the composition. Service compositions and processes are compositions; therefore, they can also exhibit any of the composition patterns.

7.2.5.3 Service compositions

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

7.2.5.4 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 4, 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 composition pattern, processes can be as follows:

- **Orchestrated**: When a process is orchestrated in a business process management system, then the resulting IT artefact is in fact an orchestration; i.e. it has an orchestration composition pattern. This type of process is often called a process orchestration.
- **Choreographed**: For example, a process model representing a defined pattern of behaviour. This type of process is often called a process choreography.
- **Collaborative**: No (pre)defined pattern of behaviour (model); the process represents observed (executed) behaviour. (See ISO/IEC 18384-3:2016, Clause 8.)

7.2.5.5 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 may need to be supported by the following:

- business process management — activities that enable discovery, modelling, execution, changing, governing, and gaining end-to-end visibility on business processes (see Reference [14]);
- business activity monitoring — activities that monitor business operations and the processes and associating SLAs and key performance indicators (KPIs).

7.2.6 Service Registration and Discovery

Service registration is the process by which service descriptions are made available to prospective service consumers. This can be accomplished by using a variety of mechanisms, including adding the service description to a collection of descriptions, storing it in a service registry/repository that provides other management and search capabilities, or making it available to another service discovery mechanism. Simple publication of documents may work fine with a small number of services, but a service registry/repository can expedite that search when there are a large number of services, as it allows a large, searchable database of services to be created and maintained.

Service registration may require the ability to add, delete, modify, classify, and verify the service descriptions, and may be maintained by service providers or others designated with the responsibility of maintaining service description.

The terms registry, repository, and catalogue are not differentiated. The single term, service registry/repository, is used to refer to this organized collection of service descriptions and other artefacts. Service description is focused on for the support of registration and discovery.

Service discovery is the process by which service consumers provide search criteria, often in terms of service description information, in order to discover services according to their specific functional or non-functional requirements. Service discovery is done during design time as well as during runtime. During the process of discovery, a service consumer may require the ability to query for and retrieve the service descriptions

A service consumer may use the service registry/repository for service discovery, typically, at design time, or at other times when the consumer needs to reference or retrieve the information contained in the service description or other service artefact reference or retrieval is needed, such as finding the service endpoint in support of late binding.

Figure 5 shows the steps illustrating a service consumer’s use of a service registry/repository.

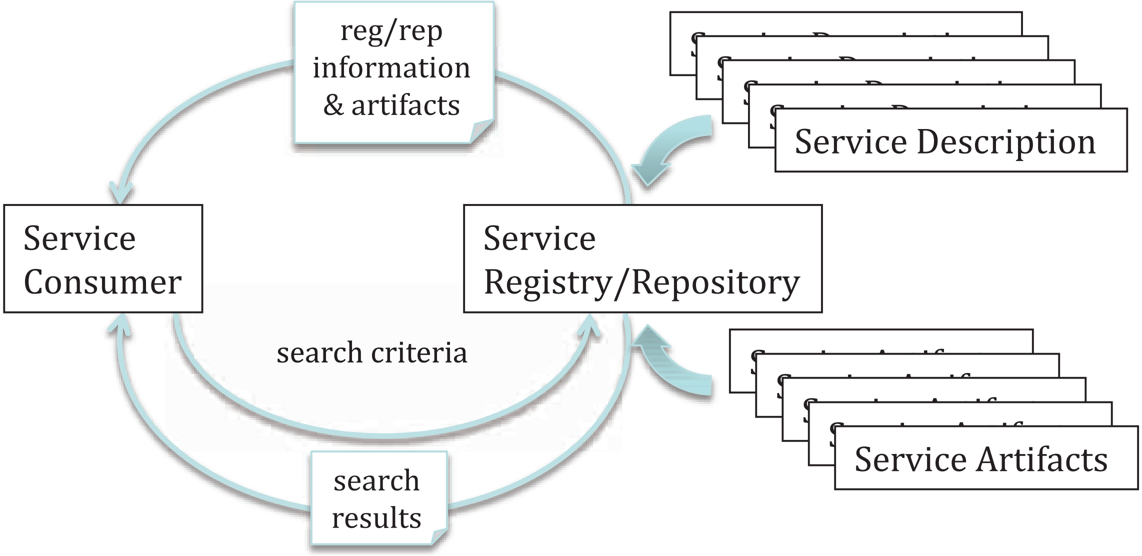

Figure 5 — Model for Service Registry/Repository Use

As shown in the example in Figure 5, the service consumer begins by submitting a search to the service registry/repository; the service registry/repository responds with search results. The search results contain information about service registry/repository contents that match the search criteria and possibly links or other mechanisms, e.g. service calls that enable the service consumer to retrieve information or artefacts from the service registry/repository. This example illustrates the service consumer as consuming both information about artefacts and the artefacts themselves.

As noted with respect to Figure 5, registry/repositories are not restricted to storing only service descriptions if they support other capabilities in the SOA solution, like information architectures, management, and governance. Architecturally, these may be considered different kinds of registry/repositories in a SOA solution, but during implementation, these registry/repositories can be collocated in the same registry/repository, leveraging the common aspects of description and discovery, storage and retrieval. The registry/repositories supporting governance can provide access to the governance policies and documentation of processes for communication and enforcement; the service descriptions can link to policies and processes that are relevant to the particular service.

Service registry/repositories are governed and are often used by the governance processes. For example, a governance process for maximizing service reuse may require solution architects and developers to search for existing appropriate services before developing new ones. Effective reuse, in this case, requires that the services are clearly described and that those descriptions are readily available to developers.

Management of registry/repositories and registration processes is recommended and may include monitoring, managing storage, backup, recovery, and notifying about exceptions and failures.

7.2.7 Service Description, Interfaces, Policies and Contracts

7.2.7.1 Service Description

A service description contains the information necessary to interact with the service and can be described in such terms as the service interface (service inputs, outputs, and associated semantics) and service policies (conditions for using the service). Service contracts may reference information in service descriptions.

The purpose of combining service interfaces and service policies in a composite service description is to facilitate interaction and visibility, particularly when the participants are in different ownership domains (see Reference [17]) for an explanation of ownership domain). Explicit service descriptions provide information necessary to know what a service does, the conditions of use, the user experience during interaction, the details of information exchange, and the location at which information exchange occurs. It makes it possible for potential participants to construct systems according to the descriptions and not the implementations.

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

7.2.7.2 Service Interfaces

Service descriptions can contain service interfaces, contract terms, and policies. Therefore, they will be defined before defining service description. Service interfaces define the way in which other elements can interact and exchange information with a service in the processing of a service request (see ISO/IEC 18384-3:2016, 7.13). Interfaces include the definition of parameters that are used for information passing when a service is invoked. This is usually documented in the service description.

The specific nature of how an interface is invoked and how information is passed back and forth differs between business 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 has 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. In practice, depending on the nature of the service and the service interface in question, these constraints may be defined either formally (documented) or informally (undocumented). Constraints should be documented and formally defined.

A single service interface can be used as an interface to 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. This also requires that the semantics of the defined interface be honoured by all services implementing the interface.

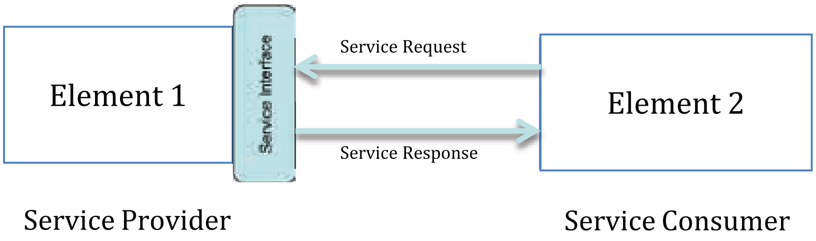

Figure 6 — Service Provider and Service Consumer using Service Interface

In Figure 6, service providers and service consumers interact via elements communicating using service interfaces, element 2 in the service consumer sends a service request to the service interface for element 1 in the service provider, and element 1 performs the appropriate functions and sends a service response (conforming to the response defined in the service interface) back to element 2.

7.2.7.3 Policy

A policy is a set of legal, political, organizational, functional or technical statements of intent or obligation for communication and cooperation. As shown in Figure 7, policy is made and asserted by a human actor and for which the human actor commits to uphold and, if desired or necessary, enforce. A human actor asserts a policy; however, a policy may provide a constraint on any actor, human or not.

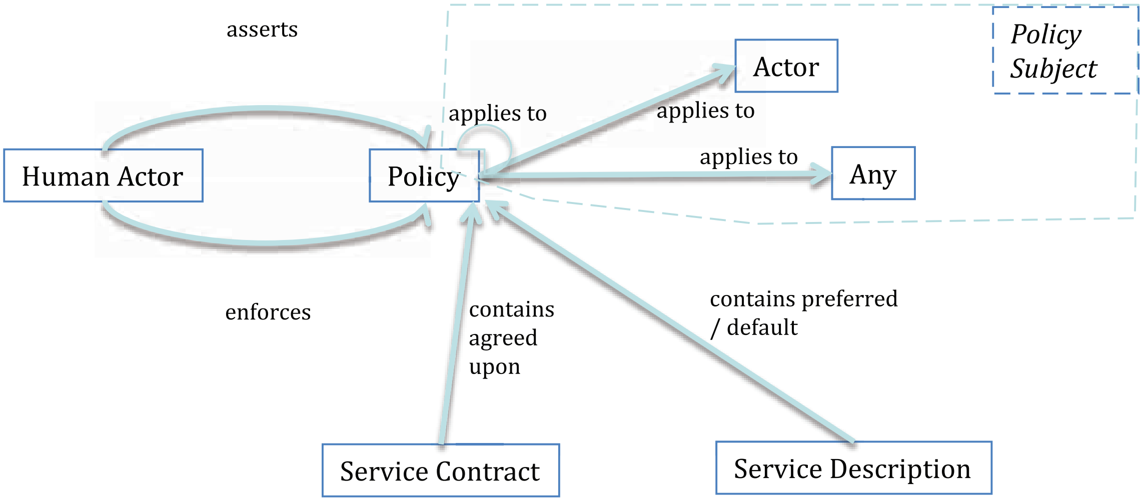

Figure 7 — Policy and Policy Subsets

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

A policy is an assertion (statement of intent) that can be stated and then applied to any number of policy subjects. For example, a policy could state that all communications with a SOA service shall be encrypted, and that single assertion is then applied to (and compliance can be assessed for) every communication with every service. An actor asserting a policy is responsible for its enforcement through direct action or delegated responsibility. Note that having a policy apply to multiple subjects does not mean that these subjects are the same, only that they are (partly) regulated by the same intent. Conversely, one or more policies may be applied to any subject. Note that where multiple policies apply to the same subject, this is often because the multiple policies are from multiple different policy domains (such as security and governance).

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 subjects to which they apply are important aspects of any system, and especially to SOA systems with their many different interacting elements.

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 (discussed in more detail in 7.2.7.3). While a policy is stated by one human actor, a contract is an agreement made by two or more human actors (the contracting parties) on a set of conditions (or contractual terms) together with a set of constraints that govern their behaviour and/or state in fulfilling those conditions. Those conditions and constraints are the basis for assessing whether the parties comply with the terms of the contract.

Policies may be stated by a service provider as the default conditions of use for a service. As shown in Figure 7, the service description is a likely place to capture such policies, possibly stating the policy in the description but more likely as a reference to an external statement of the policy. A service consumer may also state policies, and the service contract serves as a resolution of conflicts among the stated policies. A default policy can be stated or referenced in a service description because it is a constraint as stated by a single party — that service provider — and applicable to a particular policy subject — that service. A service contract, on the other hand, may be specific to the particular service provider and service consumer; it is not generally applicable to the service and may be of a proprietary nature between the service provider and service consumer. Thus, service contracts would not be included in a service description.

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 agree to, service contracts are themselves not policies because policy describes intent of at least one party and contract is the agreement between multiple parties.

7.2.7.4 Service Contracts

A service contract is the set of the terms, conditions, and interaction rules that interacting participants have agreed to (directly or indirectly). It is binding on all participants in the interaction: both logical and physical representations of service consumers, service providers, and the service itself. The service contract may include specific agreements to define how to regulate service use or to ensure that the service interactions occur in a certain sequence.

All participants interacting with a service need to comply with 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, but rather it 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 shall be created before it can be tracked.

Service contracts explicitly prescribe both the operational interaction aspects, e.g. service level agreements (SLAs), and the legal agreement aspects of using services. It is possible to split the interaction and legal aspects into two different service contracts, depending on the needs of the application. Legal agreements may 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, including, for example, who is the provider and who is the consumer from a legal obligation perspective.

A given human actor may be party to zero or more service contracts. Similarly, a given service contract may involve zero or more 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 (a.k.a. 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".

As shown in Figure 8, it is important to distinguish between a service description and a service contract. A service description captures facts about a service that are independent of any specific interaction; a service contract captures conditions specific to a certain set of service providers, service consumers, and possibly other interested third parties, such as regulators. While the facts that describe a service may be distributed among separate artefacts, the aggregated set of facts should comprise a logical consistent description. Conversely, service contracts are tailored to the participants in the service interaction, and therefore, any number of service contracts may be associated with a given service. A service description may provide default conditions for a service contract, such as default performance, whereas a service contract may specify more or less stringent performance targets.

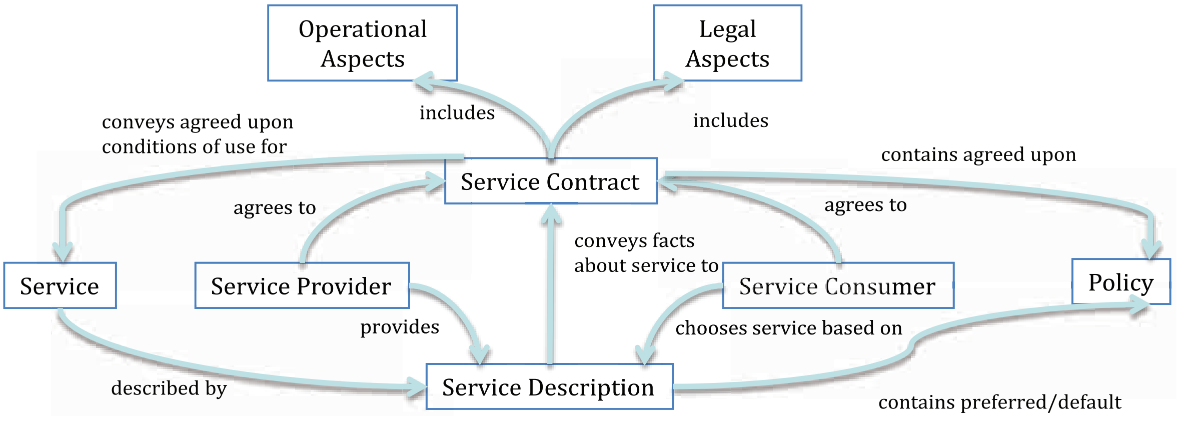

Figure 8 — Service Contract Relationships

7.2.8 Service and SOA solution lifecycle

7.2.8.1 Service Lifecycle

The service lifecycle consists of the activities, roles, and work products for just the service. These are often an extension of the organization's software development lifecycle. In contrast, the SOA solution lifecycle addresses a much larger scope, i.e. the entire service-oriented solution that provides value to the organization. Which services will be developed is determined by the SOA solution portfolio and during SOA solution lifecycle.

As shown in Figure 9 (the cycle progresses clockwise), service lifecycle can include the following.

a) Analyse and design — may include requirements management, service definition, service realization planning. The lifecycle may start with an analysis to determine what services should be in the services portfolio and drive business value, also known as identification of services.

b) Create — includes service modelling, service implementation, service testing. This includes the design of service implementation and development of service implementation code of other supporting components.

c) Assemble and deploy — may include service assembly or acquisition, service deployment, service configuration, service publication. This phase includes the assembly of components, services, and artefacts to enable deployment. After development of all these elements, the deployment of the service into a service provider's operational systems can be done. At this point, it can be instantiated so that the instance of the service is ready to interact with a service consumer.

d) Govern and manage — may include service management and monitoring, service support, service retirement, policy management.

When the service has transitioned to an operational phase, key business metrics, performance metrics, and availability should be monitored. The results of monitoring are used in dashboards, SLA enforcement, billing, governance processes, etc. During this operational phase, the service can be configured or reconfigured, fixed if it malfunctions, or improved if business requirements change.

The service lifecycle processes should be managed and governed, especially service governance, policy management, requirements management, and configuration management processes. Asset and registry/repository 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.

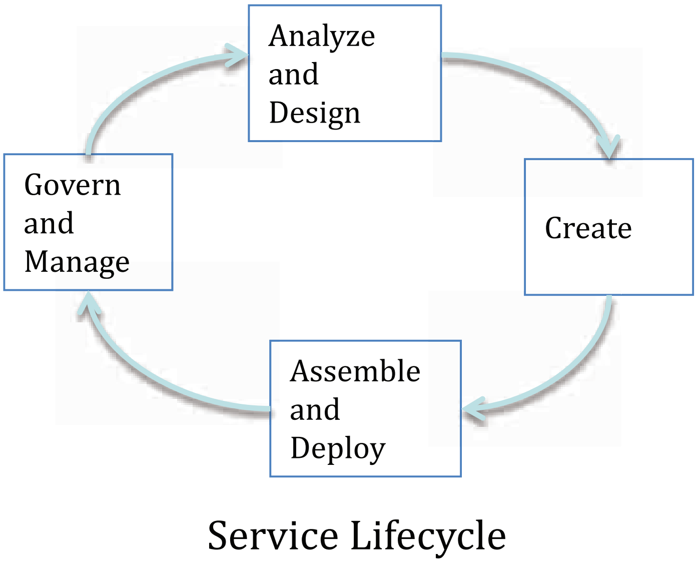

Figure 9 — Example of a Service Lifecycle

7.2.8.2 SOA Solution Lifecycle

A solution lifecycle describes the activities, roles, and work products as they relate to providing business solutions based on SOA.

The SOA solution lifecycle may begin 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 to facilitate the needs of the business.

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

As shown in Figure 10 (which progresses clockwise), four phases of the SOA solution lifecycle could be as follows.

- **Model** — Modelling is the process of capturing business design for the SOA solution 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 artefacts that will implement the business design for the SOA solution. The business design can be 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 the SOA solutions and the actual deployment of those solutions. This includes resolving the SOA solution's resource dependencies, operational conditions, capacity requirements, and integrity and access constraints. In addition, the techniques for enabling availability, reliability, integrity, efficiency, and serviceability should be considered for the SOA solution.
- **Govern and manage** — Govern and manage considers how to maintain the operational environment and the policies expressed in the assembly of the SOA solutions 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.

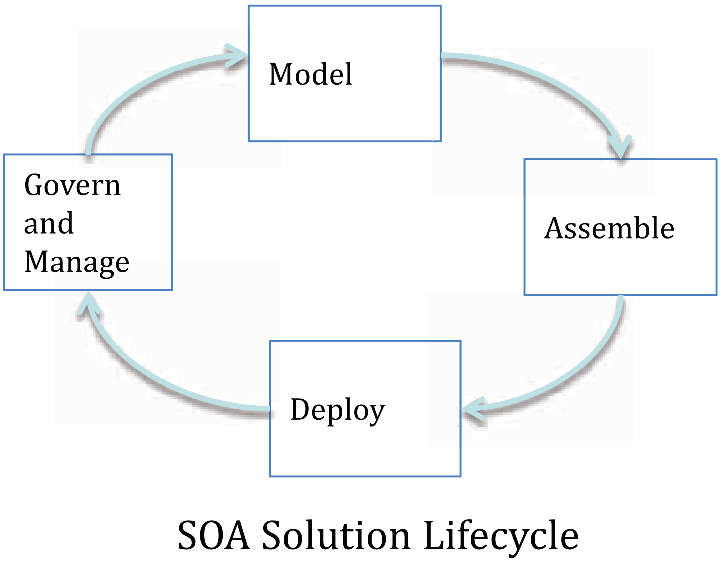

Figure 10 — SOA Solution Lifecycle

Progression through the lifecycle is not entirely linear. For example, 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. Information technology constraints established in the assembly phase may limit the business design created during the model phase.

7.2.8.3 Services and SOA solution lifecycles relationship

The ideas around SOA services and SOA solutions are complementary and intertwined. Typically, we look to SOA services as being the building blocks of SOA solutions, but those solutions offer services and are combined into other solutions. Clearly, the two lifecycles are closely related; the SOA solutions lifecycle provides a backdrop for the SOA services lifecycle. The two lifecycles follow the general phases for software, i.e. requirements analysis, design, development, deployment, operations, and retirement. However, the target of each of those phases is very different. Services lifecycle is focused on just service itself, where the solution covers business aspects and the interrelationships between the services, operational systems, and supporting aspects. As shown in Figure 11, just as solutions are a backdrop for services lifecycle, the governance vitality method is a backdrop for solutions lifecycles.

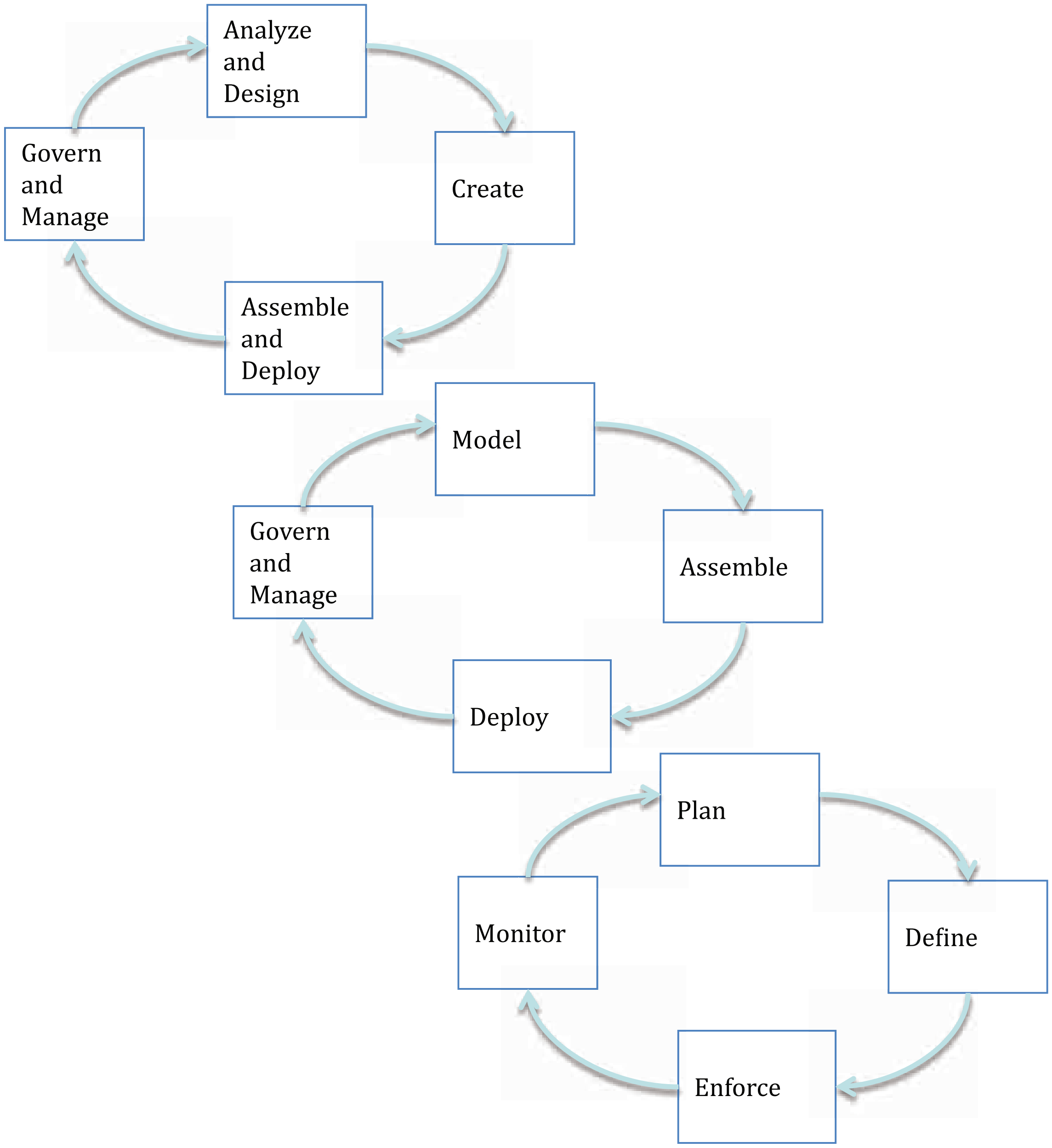

Figure 11 — Relationship between Service, SOA Solution, and SOA Solution Governance lifecycles

As with any lifecycle, roles, responsibilities, and work products need to be defined. The use of SOA principles and concepts implies

- a continuing relationship between the roles of service provider and service consumer,
- accurate and updated service and solution descriptions,
- sufficient metrics to have a clear understanding of performance and setting of expectations as overall use grows and evolves, and
- modelling of the SOA ecosystem to predict and respond to growth and evolution.

An important consideration as one progresses through the lifecycle is that the services and solutions exist within the SOA ecosystem and may concurrently support the needs of many service consumers. Thus, while individual performance issues are important and should be considered, when the focus is on optimizing the whole, then trade-offs may need to be made so that the individual services or solutions may run sub-optimally. While understanding local optimization is important, a SOA approach will introduce more complex trade-offs that make modelling more essential.

7.2.9 Loosely coupled

In general, loose coupling is accomplished by reducing dependencies between elements in a system.

For example, loose coupling for services is improved when service consumption is insulated from the underlying implementation, usually via the use of service interfaces, access policy, and the ability to discover services. Additionally, loose coupling of the service interface can be improved by exchanging semi-structured documents as a whole. Often, this is more tolerant of changes in the details of the information exchange than procedure call style exchanges.

When the service consumer fully relies on the standards based service description that serves as the "contract" between the consumer and the provider, more insulation is provided. These types of insulation removes the dependencies of programming languages, platforms or any special provider conventions such as time sequencing, address binding, access language or access protocol.

The service oriented architecture (SOA) architectural style supports loosely coupling the interface of a service being consumed from the implementation being provided. The message exchange should be well-defined but still provide principled mechanisms for variations in inputs and extensibility.

Loose Coupling represents the allowance for variability and flexibility and enables reusability and robustness when combined by service orientation.

7.3 Cross Cutting Concerns

7.3.1 Defining Cross Cutting

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 concern because it applies to many areas, including operational systems, services, business processes, and service consumers. All of these need to be managed, but how these are managed is different based on the management target. Managing infrastructure like storage and databases 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 cross-cutting aspects for SOA are as follows.

7.3.2 Integration

Integration is a central aspect of the design and development of SOA solutions since there are often cross-service and cross-solution domain interactions to consider. Complex SOA installations can reach across business unit and partner boundaries, and integration is used to leverage existing services, as well as incorporating new services across service domains. Integration aspects and supporting technologies need to be identified even when service interactions are simpler and within a domain.

7.3.3 Cross Domain interaction

Service discovery and management may need to span various solution domains in an organization or across organizations. 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.

Figure 12 is an example that shows a number of solution domains, each implemented with its own technology, where the interactions between them need to be managed as a whole.

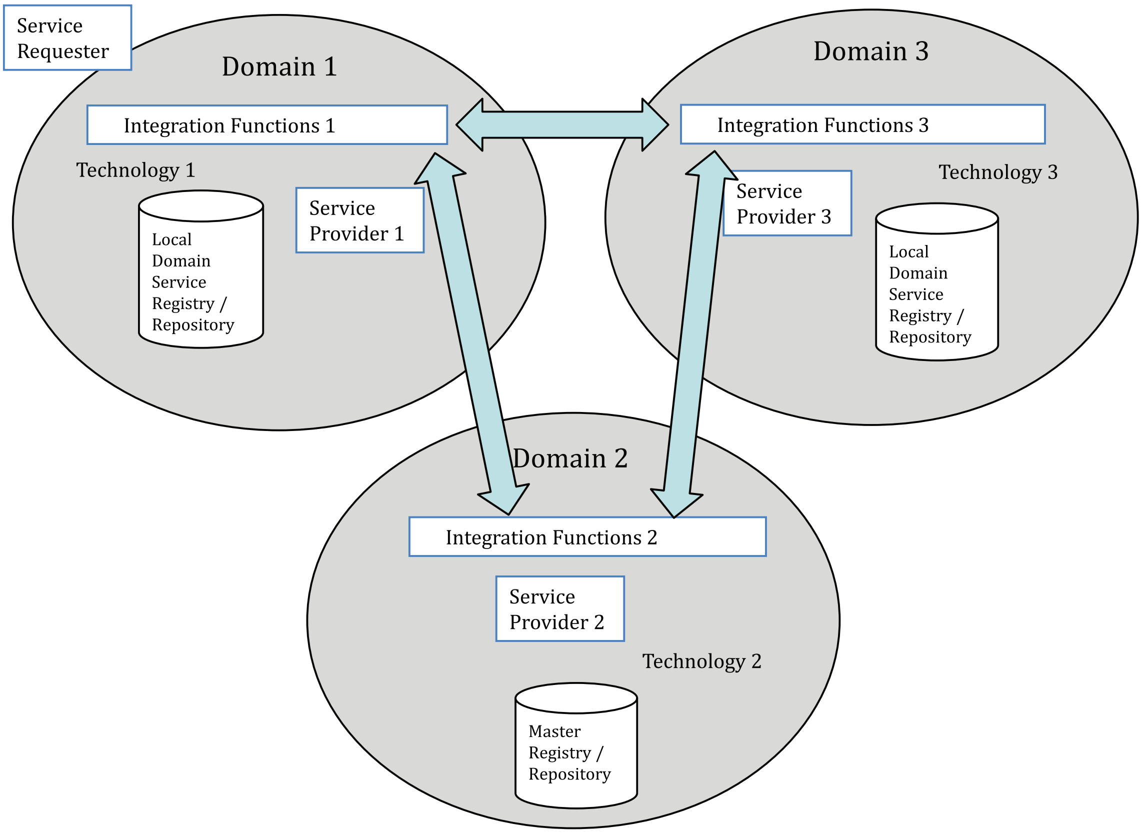

Figure 12 — Cross Domain Interaction

Both single solution domain SOA infrastructure and heterogeneous federated SOA infrastructure address the management of cross cutting aspects, especially of service discovery, service management, service security, and service governance.

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

7.3.4 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. Service integration may include an intelligent routing, protocol switching, and interface transformation mechanisms as is often seen in a service bus. The integration functionality in Figure 12 enables the loose coupling between the request and the concrete provider by matching the service request and service implementation.

This loose coupling 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 organizations drives the need for a more flexible management of the services and the corresponding artefacts. Three aspects of integration that need to be considered are transport, transformation, and mediation.

— **Transport** — A message originating with a sender should be able to reach its intended receiver. This is accomplished through use of a transport infrastructure that provides the protocols necessary to convey messages between the two. The sender and receiver may both be connected to a common transport infrastructure or bridging will be necessary between their respective infrastructures. This bridging may require protocol translations, as well as the need to provide routing and correlation of messages between the endpoints.

- **Transformation** — The sender and receiver should have a common understanding of the message content. This may require a transformation between formats for presentation of content or between vocabularies in order to align local semantics. An example here is where one endpoint uses a different version of a data format with extra fields, so the extra fields need to be removed in order for the message to 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.

7.3.5 Management and Security

7.3.5.1 General

The management and security concern is an important aspect for SOA and provides ways of dealing the non-functional requirements (NFR) in any given solution. This provides the technical means to enable a SOA solution to meet its requirements with respect to: monitoring, reliability, availability, manageability, transactionality, maintainability, scalability, security, safety, life cycle, etc. It includes the scope of the traditional FCAPS (fault, configuration, accounting, performance, security) from ITIL or RAS (reliability, availability, serviceability).

Common aspects of management that may need to be supported and managed are transactions, availability, service reliability, metadata management, and management services.

- 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. Two types of transactions are atomic (which are usually short lived and rollback upon failure) and business activity transactions (which are usually long running with defined compensations in case of failure). See Annex B for more information on transactions in SOA solutions.
- Availability includes both SOA service availability and SOA solution availability. 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 tied to a provider, network, and consumer boundary, checking that the service, system, and network are functioning. Service availability is different from different boundaries. 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.

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 service consumer and service provider, it does not address whether a request is actually processed or not by the provider. Transactions address whether a request is processed or not.

a) Reliability, in the context of 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 service consumer to service provider. The four reliability qualities for systems are: at-least-once, at-most-once, exactly-once, and ordered. For more information on these, see Annex B. The reliability required for the SOA solution and service interactions will affect key architectural decisions. For example, where is it not possible to design an idempotent service (see Annex B), other reliability qualities may be requested, notably at-most-once and exactly-once.

b) 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. 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 can be implemented directly by services to enable manageability.

c) 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.

7.3.5.2 SOA solution 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 has a broader scope and manages the entire SOA solution, or set of SOA solutions.

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.

Common types of management used for SOA solutions are as follows.

— Service and SOA solution 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. For SOA, this can be challenging when the services being managed or the solution uses services that are in another ownership domain, either in the organization or cross organization. Metrics per consumer of the service need to be negotiated, collected and made available, often on a dashboard. For cross domain services, delivery of that information may be delivered via reports, events, or other information feeds into their own dashboards.

— Business activity monitoring and management: provides monitoring and management of business activities, business events, and business processes. It provides ability to analyse event information (real-time and near real-time), as well as stored (warehoused) events. It enables review and assessment of business activities which determines responses and issues alerts/notifications. This leverages the support in the organization for service and SOA solution management and event management. Dependencies across organizations should be managed appropriately.

— 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 includes management of virtualized systems. For SOA solutions, these systems may be represented and accessible as services themselves and the resources may be within the organizations control or across domains. For cross domains, reporting to service consumers and customers may require special negotiation and enablement.

— Security management: provides monitoring and management of security and secure solutions. This includes the ability to manage roles, identities, access rights and entitlements, as well as protect unstructured and structured data from unauthorized access and data loss. Security management should also address how software, systems, and services are developed and maintained throughout the software lifecycle. It should maintain the security status through proactive changes reacting to identified vulnerabilities and new threats and enable the IT organization to manage IT related risks and compliance. These capabilities provide the basis for automation of security management. For SOA, security policies for negotiating across organizational domains need to be clearly defined in the descriptions and supported.

- Policy monitoring and enforcement: provides mechanism to monitor and enforce policies and business rules for the SOA solution and services. This includes finding and accessing policies, evaluating and enforcing policies at check points and enforcement points. Policy enforcement includes enforcement when metrics are captured, signalled, and recorded. Enforcement may send compliance status, metrics, notifications, and log of non-compliance. Notifying across organizations when policies have been breached should be supported. In addition, escalation procedures for policies that have not been met should be defined up front and enabled operationally.

- Event management: provides the ability to manage events and enables event processing, logging, and auditing. For SOA, using events between organizations is a common pattern of notification from dashboards, monitoring and management systems. While auditing should always be enabled, audits are especially important for cross organization solutions.

- Configuration and change management: provides ability to change solution and service configuration and descriptions. Configuration and change management may need to be coordinated across organizations for particular services and SOA solutions.

- Solution and service lifecycle management: provides mechanisms to deploy, start/enable, stop/disable, and un-deploy services and SOA solutions. For organizations that use services owned by other domains or organizations, the lifecycle of the services in the portfolio will need to be coordinated. Mechanisms to communicate when maintenance or updates on services are needed or when the service consumer no longer needs the service.

- Service metadata management: provides management of additional service metadata and physical documents such as the service descriptions and policies related to the services. Metadata can be used to enable many of the SOA characteristics: i.e. discovery, loose coupling, integration, as well as service relationships and properties.

7.3.5.3 SOA security

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

SOA security should follow the industry best practices, such as the recommendation developed by ITU-T on X.805 and the dimensions covered by the WS-Security (see Reference [28]) set of standards.

The threats that SOA security needs to address are destruction, corruption, removal, disclosure, and interruption (see Annex B for more information on these threats).

The eight security dimensions that are required to protect against the threats listed are access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy (see Annex B for more information on these dimensions for SOA solutions).

A few key concepts that have special concerns for SOA solutions are as follows.

- Authentication. Authentication should provide a proof of identity for the use of services or any component in the SOA using, for example, shared secret, PKI, or digital signatures. It should be noted that as SOA addresses interactions between service consumers and service providers, the proof of identity should be carried between the initial consumer and the ultimate provider. In service choreographies, there will be multiple actors that may all have to be authenticated. When services are offered between two or more entities that are organizations, there may be a risk that the churn of actors leaving or joining these entities will make the environment very difficult to manage. Using federated identities that implement a trusted model between entities should be considered for SOA solutions where there is propagation of assertion tokens that provide consistent authentication of the initial requester through a trusted protocol between components (an example is WS-Federation, see Reference [29]).

- Non-repudiation. Non-repudiation should prevent the ability to deny that an activity on components or elements in the SOA solution occurred. Examples include system logs and 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.
- Availability. Availability should ensure that elements, services, and components are available to legitimate users. This should be coordinated with the SLAs and other contract metadata that specify the conditions of use of the service.

7.3.6 SOA Solution Governance

Governance is the establishing and enforcing of how people and solutions work together to achieve organizational objectives. For an organization implementing SOA principles, SOA solution governance should assist in setting expectations and providing a means to reduce risk, maintain business alignment, and show business value. Like other kinds of governance, the role of SOA solution governance is to create a consistent approach across services, processes, standards, policies, and guidelines while putting control and compliance mechanisms in place. This focus on putting controls in place distinguishes governance from day to day management activities (see Reference [10]).

Some of the common reasons that SOA drives the need for, and sometimes cannot be successful without, governance are as follows.

- SOA is often driven by organizational changes.
- SOA often drives organizational changes.
- SOA is implemented across organizational boundaries, especially in the sharing of services.
- Reuse of services often requires organizations to create new cost-sharing requirements.
- SOA drives new practices in an organization for identifying services and sharing services.

Annex A summarizes the ISO/IEC 17998 SOA governance framework technical standard, providing details of a SOA governance reference model, a SOA governance vitality method, and best practices for creating a governance regimen for services and SOA solutions.

In effect, SOA solution governance extends IT and EA governance. This implies 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.

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

a) processes — including governing and governed processes. These are unique to SOA;

b) organizational structures — including roles and responsibilities;

c) enabling technologies — including tools and infrastructure.

In order to specify the changes necessary to accommodate SOA in an existing governance regime, these governance activities are 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. It should include governance activities that are impacted by SOA.

Governing processes, compliance, dispensation, and communication, realize the intentions of the organization. The processes and activities being governed for SOA solutions are specific to the planning, design, and operational aspects of SOA. As described in Annex A, SOA solution governance should consider the following.

— 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.

Since SOA solution governance requires an on-going process to maintain alignment, therefore, it is unrealistic to expect an organization to define and deploy in a single project an organization-wide SOA solution governance regimen that meets its long term aspiration as goals for SOA. It is expected to evolve as an organization's SOA experience grows. To support this, a SOA governance vitality method takes a particular organization through a number of phased activities that form a continuous improvement loop, defining a roadmap for deploying governance. It is a continuous process in which progress is measured, course-correction is made, and updates are performed

8 SOA Architectural Principles

8.1 Architectural Principles defined

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

— Location independence: there is no preferred location for service consumers and service providers. They could transparently both be located on the same system, or in different organizations and in different physical locations.

— Implementation independence: there is no requirement for specific platform or implementation technologies for service consumers and service providers to adopt. They should not need to be aware of the other party's technical environment or implementation details in order to interoperate.

— Protocol independence: there is support for services to 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, service buses may perform the connection between heterogeneous services.

Services should be designed to be: interoperable, described, reusable, discoverable, composable, autonomous, loosely coupled, and manageable. Each of these characteristics is described in this Clause using a list of activities that architects should consider performing in order to architect and develop services and SOA solutions that exhibit these characteristics. Some of the activities apply to multiple characteristics. In addition, each characteristic includes a list of characteristics or business benefits that the service or SOA solution may exhibit as a result of satisfying the characteristic.

8.2 Interoperable — Syntactic, semantic

The characteristic of interoperability is achieved when people, organizations and systems are able to communicate, exchange information and effectively use the information to work together.

Interoperability has multiple dimensions, and is often viewed from a systems engineering perspective including consideration of technical, syntactic, and semantic factors. A broader view of interoperability may include social, organizational, legal, political, and other factors.

Technical interoperability involves the use of common methods and services for the access, communication, storage, and processing of data, usually in the applications platform and communications infrastructure domains, based on standards and/or common IT platforms.

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. Syntactic interoperability is a pre-requisite to achieving further interoperability.

Semantic interoperability can be achieved through establishing a shared understanding of intent, allowing recipients to act on the contents of messages in a manner consistent with the sender's intent. In other words, there should be a shared understanding of the business requirements being addressed by the service, not just an agreement on common syntax.

Interoperability is facilitated by

- defining protocol definitions that include the:
  - 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 interactions, e.g. call backs;
  - definitions of any restrictions on invocation order, e.g. open before write;
- defining polices that affect reliability, availability, security, and transactionality.

The benefits of interoperability are it

a) includes a minimum level of semantic understanding of the services being consumed and provided,

b) improves understanding and collaboration across system and organizational boundaries,

c) enables increased likelihood of two systems interacting correctly,

d) enables a decrease in erroneous interactions, and

e) enables reduction in ambiguity when errors and faults occur.

8.3 Described

The characteristic of being described is achieved when the service consumer, service provider, or developer can find, access, interpret, and process a description of a service. The service description, as described in 7.2.7, is metadata that may include details on the service contract, service interface, and policies for both service consumers and service providers.

Being described is facilitated by

- capturing the service interaction contract in a standardized format, e.g. XML, WSDL (Web services definition language, see Reference [18]) for Web services implementations,
- storing the descriptions in the registry/repository so that it can be located, accessed, and reused,

- isolating configuration into policy descriptions,
- capturing information for service level agreements in descriptions,
- capturing QOS available in descriptions,
- capturing business, governance, and management processes that use the services in descriptions, and
- capturing service location.

The benefits of being described are it

a) enables loose coupling and late binding by providing sufficient description to be able to look up and bind to the service instance at runtime,

b) enables reuse by adding metadata so that the appropriate services can be located and reused,

c) enables reuse by adding description so that the appropriate services are available for the functions needed,

d) enables management and reconfiguration of services at runtime and when being reused to minimize changes to the service implementation,

e) enables management of services at runtime by describing management monitoring and processes,

f) enables business and IT governance of SOA solutions and services, and

g) enables addition of an integration functionality (see Figure 10), service bus (see 2.22).

8.4 Reusable

The characteristic of reusability is achieved when the same service definition or implementation or both, are used in more than one SOA solution. Enabling reusability helps to reduce development and maintenance costs and increases the long-term return on investment for the solution.

Reusability of services is facilitated by the following:

- 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;
- governing 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.

The benefits of reusing services are it

a) enables reducing cost of development by eliminating the need to develop a set of services,

b) enables reducing cost of SOA across the organization over time since services do not need to be redeveloped for multiple solutions now and in the future,

c) enables decreasing development time to value since you are using existing services that do not have to be developed,

d) enables decreasing risk since the services have been well-tested before being reused,

e) enables increasing agility where the service can be used in unanticipated ways, and

f) enables increasing agility since the services are available for rapid development.

8.5 Discoverable

The characteristic 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.

The ability to locate or discover services and their descriptions based on specific criteria can be enabled by a registry/repository or other kinds of search technologies. Service consumers should fundamentally be able to find services and be able to interact before the rest of management, security, and governance are needed.

Discovery infrastructures exist on a continuum, from being provided a service description (no discovery at all) to looking in a well-defined location or directory to sophisticated, searchable registry/repositories.

Services may be discovered at design or runtime in the service lifecycle. Supporting discovery during runtime supports late binding for the services which increases loose coupling.

Connections also exist on a continuum from a tightly coupled static direct interaction where the service information could be provided from a-priori knowledge directly into the service implementation to dynamic location of candidate services, selection of those services via metadata, and binding to the right service at runtime. In between is the common pattern of discovery of the service description during design time from a registry/repository and use during modelling, assembling, and deploying, and the lookup and binding of the service address at runtime. Discovering the endpoints for the services at runtime can increase loose coupling.

Discoverability is facilitated by

— storing service description artefacts in a registry/repository or some other predictable location,

— providing search facilities/tools to be able search for, find, and access the service description, and

— having governance processes dictating registration and storage of service descriptions.

The benefits of discoverability are it

a) enables late binding to services, which increases robustness,

b) enables loose coupling by providing location and access to the service description,

c) enables reuse of services by ensuring that existing, reusable services and service descriptions can be located and used, and

d) enables dynamic composability.

8.6 Late Bind-able

The characteristic of late binding uses accessing information at runtime to set up the binding and can be independent from discovering the service

Using late binding via a registry/repository also allows the service consumer to be isolated from service location changes. Combining use of policy with late binding further isolates the service consumer from the service implementation by allowing configuration of the service interaction. Finally, adding the use of an integration functionality isolates the service consumer and the service implementation and enables an increase in 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.

Discoverability and late binding can help enable dynamic composability.

Late binding is facilitated by

- providing search facilities/tools to be able search for, find and access the service description, and
- governing processes supporting the use of binding to the final endpoint at runtime.

The benefits of late binding are it

a) enables increased robustness,

b) enables loose coupling by providing location and access to the service description,

c) enables increasing robustness and agility of the SOA solution,

d) enables reuse of services by ensuring that existing, reusable services and service descriptions can be located and used, and

e) enables dynamic composability when enabled using service registry/repositories.

8.7 Composable

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

The capability to compose a service is enabled by appropriately described services that are loosely coupled and autonomous.

Composability is independent from the types of compositions that can be created, such as processes, choreographies, and orchestrations.

Composability is facilitated by

- adhering to the principle of reuse,
- developing autonomous services, and
- discovering services via artefacts in a registry/repository.

The benefits of composability are that it

- enables agility by the reuse of existing services to develop new services without waiting for development, and
- enables the development of compositions, processes, choreographies, and orchestrations.

8.8 Self-Contained

The characteristic of self-containment, or autonomy, 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 should be encapsulated and should not depend on other services for their state or are stateless. Services that are compositions of other services can be self-contained as long as they do not break encapsulation or allow service consumers to see or understand how the services are implemented.

Self-containment is facilitated by

- isolating the service consumer from the implementation (opacity),
- developing self-contained services,
- developing complete descriptions, and
- providing an implementation-independent interface.

The benefits of self-containment are it

a) enables increasing reusability, and

b) enables increasing composability.

8.9 Loosely coupled

The characteristic of loose coupling is achieved when service consumption is insulated from underlying implementation.

In most industries, anything that is reusable allows some variability and flexibility when connecting to other elements. 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. The message exchange should be well-defined but still provide principled mechanisms for variations in inputs and extensibility.

Loose coupling is facilitated by

- requiring service 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,
- separating implementation from both its binding and service endpoint interface,
- choosing document style interfaces rather than RPC,
- having service consumers use late binding (at runtime) to service instances via service discovery, which may be provided by a registry, repository, mediation, or other mechanism,
- externalizing configuration and runtime contracts into policy, and
- using integration functionality (see Figure 10) to provide support for connections, protocol mediation, security and other qualities of service. This relieves the service 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 it

a) enables minimizing changes to consumers of services over time, even when versions change or changes are needed for qualities of service or protocol support, given the service is otherwise acceptable,

b) enables substitution of an alternate service given the effects, semantics, and policies are the same,

c) enables minimizing changes to services by the provider to change versions, protocols, capacity, and qualities of service,

d) enables minimizing changes to service implementations to support reuse of services in new opportunities, increasing business agility,

e) enables increasing service availability and reliability for service consumers by allowing selection of appropriate service with late binding at runtime,

f) enables decreasing cost (via enabling reuse of services),

g) enables the addition and benefits of an integration functionality,

h) enables federation of SOA domains — sharing of services from one SOA solution domain to another,

i) enables management of services to be policy driven, and

j) enables reuse of services by isolating configuration into policy.

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

8.10 Manageable

The characteristic 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.

Manageability is facilitated by

- defining and monitoring key 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,
- developing governance policies and processes, and
- enabling control over the lifecycle of the service (enable, disable).

The benefits of manageability are it

a) enables awareness of service availability,

b) enables reducing risk of service failure,

c) enables contract enforcement,

d) enables governance process execution and automation,

e) enables alignment of business and IT requirements for the SOA solution, and

f) enables automation of service and SOA solution management.

Annex A
(informative)

SOA Governance Framework

This Annex is a summary of ISO/IEC 17998 SOA governance framework technical standard. It provides context for readers who need more understanding about SOA governance, service governance, and a SOA governance regimen.

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 implies 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** ("EA governance") is the practice and orientation by which enterprise architectures and other architectures are managed and controlled at an enterprise-wide level. See Reference [16].
- **Governance of IT** ("IT governance") is 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. See ISO/IEC 38500:2008, 1.6.3.
- **Business governance** ("corporate governance") is the set of processes, customs, policies, laws, and institutions affecting the way an organization is directed, administered, or controlled.

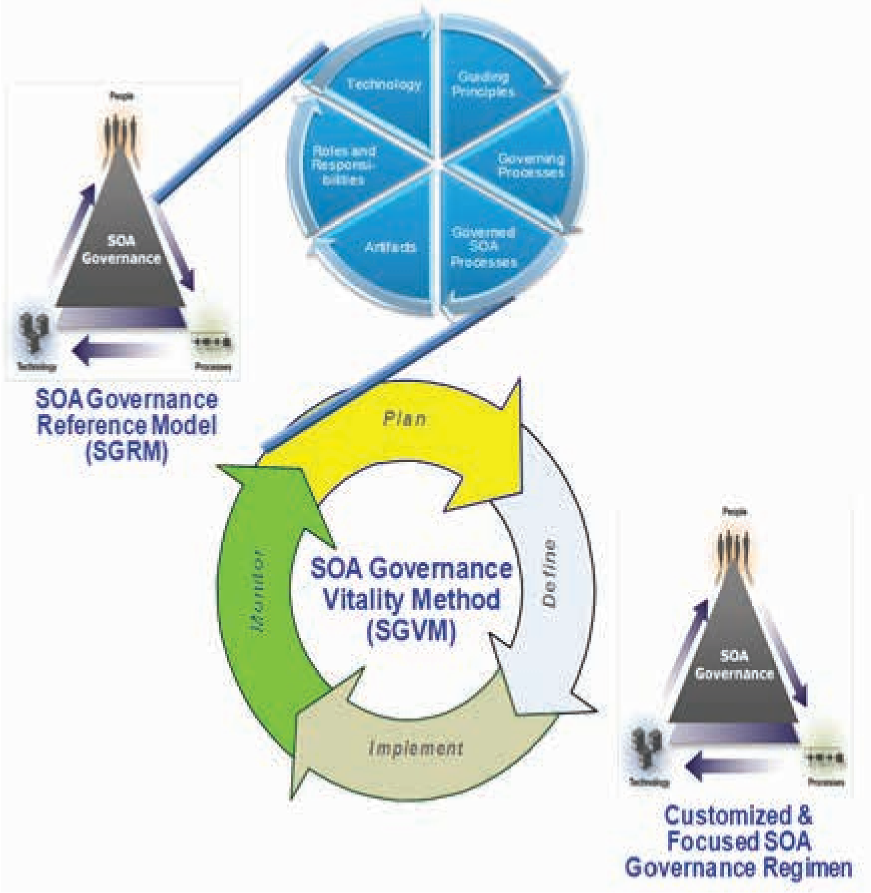

Figure A.1 — SOA Governance Regimen

Governance regimen for SOA solutions includes governance of services, as well as governance of the SOA solution in its entirety. Governance is applied to the following:

a) **processes** — including governing and governed processes;

b) **organizational structures** — including roles and responsibilities;

c) **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 need to 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 should 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. The three types of governing processes are as follows.
    - 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 solution 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** — determined via an analysis of the kinds of people involved in the activities being governed and their degrees of responsibility for those activities;
- **artefacts** — a set of artefacts 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 the following:

a) **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;

b) **define**: creates a tailored SOA governance regimen from the SOA governance reference model,[1] using the results of the plan phase. It then analyses the differences between this regimen and the existing governance regimen to create transition plans;

[1] See http://www.opengroup.org/soa/source-book/gov/sgrm.htm.

c) **implement**: realizes the tailored SOA governance regimen that was created in the define phase to produce a SOA governance regimen for the enterprise; essentially it executes the transition plans from the define phase;

d) **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.

Annex B
(informative)

Management and Security Concerns

B.1 General

This Annex expands on the cross-cutting concern of management and security and provides context and additional information regarding transaction management, service reliability, SOA security, and SOA security governance.

The management and security concern 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 management and security that need to be supported are transaction management, availability, and service reliability.

B.2 Transaction management

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 (see Reference [25]) which is the framework for WS-AtomicTransaction (see Reference [26]), and WS-Business Activity (see Reference [27]).

- **Atomic transactions** — 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 should 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.

B.3 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 service consumer and the service 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.

B.4 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 service consumer to service 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 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, it is important that the order is preserved in the request ordering if it is required between service consumer and service 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 service consumer and service provider; it does not address whether a request is actually processed or not by the provider. Transactions address whether a request is processed or not.

B.5 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 has a broader scope and manages the entire SOA solution, or set of SOA solutions. Common types of management used for SOA solutions are as follows.

- **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 solution 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 analyse this event information, both in real-time/near real-time, as well as stored (warehoused) events, and to review and assess business activities in the form of event information and determines responses or issues alerts/notifications.
- **Security management**: manages and monitors security and secure solutions. This provides ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, 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 also sends 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 un-deploy 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.

B.6 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. For example, since the business design can be captured as a model and used to assemble the solution 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.

B.7 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 functions, 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 (or ESB), and service federation.

B.8 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 and the dimensions covered by the WS-Security set (see Reference [28]) of standards.

The following are threats that SOA security needs to address:

— **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 are the following:

a) **Access control**: limit and control access to services and elements using control tokens, components and elements: password, access control lists, firewall.

b) **Authentication**: provide a proof of identity for the use of services or any component in the SOA using, for example, shared secret, PKI, or digital signatures. It should be noted that as SOA addresses interactions between service consumers and service providers, the proof of identity should be carried between the initial consumer and the ultimate service provider. In service choreographies, there will be multiple actors that may all have to be authenticated. One aspect that SOA has to manage is when services are offered between two or more entities that are organizations, and there is a risk that the churn of actors leaving or joining these entities will make the environment very difficult to manage. This is why federated identities that implement a trusted model between entities are commonly used across SOA, with the propagation of assertion tokens that provide consistent authentication of the initial requester through a trusted protocol between components. An example is WS-Federation (see Reference [29]).

c) **Non-repudiation**: prevent ability to deny that an activity on components or elements in the service oriented architecture occurred. Examples include: system logs and 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.

d) **Data confidentiality**: ensure confidentiality of data exchanged in service interactions or managed by components of the architecture. In order to avoid unnecessary disclosures, the confidentiality may have to be carried between the initial service consumer and the ultimate service provider with no disruption. An example is protecting messages using encryption with XML-Encryption.

e) **Communication security**: ensure information only flows from expected source to expected destination. Examples: using HTTPS, VPN, MPLS, L2TP. In a SOA architecture, an application or solution level validation on component boundaries may become necessary to control incoming and outgoing sources and destinations.

f) **Data integrity**: ensure that data are 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 either end of the service interaction to avoid opening a breach in any of the various nodes that may constitute the architecture between the service consumer and provider.

g) **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.

h) **Privacy**: ensure identification, service use, and service management are kept private.

B.9 SOA Security Governance

SOA security governance is a specific domain of governance that 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 processes may include the definition and exchange of certificates or public keys between trusted actors before enabling other actors in the entities to interact.

It should be noted that confidentiality and non-repudiation may cause an increase in message size and volume due to the required additional information, such as certificates and additional processing time to manage signatures.

Bibliography

[1] ISO/IEC 2382, *Information technology — Vocabulary*

[2] ISO/IEC/IEEE 15288, *Systems and software engineering — System life cycle processes*

[3] ISO/IEC 12207, *Systems and software engineering — Software life cycle processes*

[4] ISO/IEC/IEEE 42010, *Systems and software engineering — Architecture description*

[5] ISO/IEC 10746-2, *Information technology — Open distributed processing — Reference model: Foundations — Part 2*

[6] ISO/IEC JTC 1/SC 38 N0043, *Research Report on China's SOA Standards System*

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

[8] 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

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

[10] ISO/IEC 17998, *Information technology — SOA Governance Framework*

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

[12] ISO/IEC 16500-8, *Information technology — Generic digital audio-visual systems — Part 8: Management architecture and protocols*

[13] ISO/IEC 38500:2008,[2)]*Information technology — Governance of IT for the organization*

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

[15] ISO/TR 9007, *Information processing systems — Concepts and terminology for the conceptual schema and the information base*

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

[17] OASIS. *Reference Architecture for SOA Foundation,* Version 1.0, OASIS Committee Specification, 4 December 2012: see http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-v1.0-cs01.pdf

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

[19] 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

[20] 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

[21] 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

2) Revised by ISO/IEC 38500:2015.

[22] 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

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

[24] Garrett J.J. *A New Approach to Web Applications,* Feb 18, 2005 see http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[25] 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

[26] 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

[27] 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

[28] Web Services Security (WS-Security) Version 1.1 OASIS Standard, Feb 1, 2006, http://docs.oasis-open.org/wss/v1.1/

[29] Web Services Federation (WS-Federation) Version 1.2 OASIS Standard, May 22, 2009, http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.doc

[30] ISO/IEC 20926, *Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009*

[31] ISO/IEC 19505-2, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure*

ICS 35.100.05

Price based on 51 pages