TECHNICAL REPORT

ISO/IEC TR 30102

First edition
2012-12-01

Information technology — Distributed Application Platforms and Services (DAPS) — General technical principles of Service Oriented Architecture

Technologie de l'information — Plate-formes et services d'applications distribuées (DAPS) — Principes techniques généraux de l'architecture orientée services

Reference number
ISO/IEC TR 30102:2012(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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.

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

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

In exceptional circumstances, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide to publish a Technical Report. A Technical Report is entirely informative in nature and shall be subject to review every five years in the same manner as an International Standard.

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

ISO/IEC TR 30102 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology,* Subcommittee SC 38, *Distributed application platforms and services (DAPS)*.

Introduction

Service Oriented Architecture (abbreviated SOA) is an architectural style that supports service orientation and is a paradigm for business and IT (see 3.1.40). This architectural style is for designing systems in terms of services available at an interface and the outcomes of services. A service is a logical representation of a repeatable business activity that has specified outcomes, is self contained, may be composed of other services and is a “black box” to consumers of the service (see 3.1.14).

To enable this co-operation and collaboration business-oriented SOA takes ‘service’ as its basic element to constitute and integrate information systems so that they are suitable for a wider variety of application requirements. Some of the benefits of using SOA are improvement in the efficiency of development of information systems, efficiency of integration and efficiency of re-use of IT resources. It also enables agile and rapid response of information systems to ever-changing business needs. Many companies across many industries world-wide have developed SOA enterprise architectures, solutions and products.

This report is intended to be a single set of SOA technical principles, specific norms, and standards for the world-wide market to help remove confusion about SOA, improve the standardization and quality of solutions, as well as promote effective large-scale adoption of SOA. The benefits of this technical report contribute to improving the standardization, interoperability, and quality of solutions supporting SOA.

This technical report defines the basic technical principles and reference architecture for SOA rather than being focused on the business aspects. It also discusses the functional, performance, development, deployment, and governance aspects of SOA. This technical report can be used to introduce SOA concepts, as a guide to the development and management of SOA solutions, as well as be referenced by business and industry standards.

This technical report includes the following clauses:

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

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

Clause 5 – SOA Technical Framework – documents an overview of a reference architecture for building SOA based solutions.

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

Information technology — Distributed Application Platforms and Services (DAPS) — General technical principles of Service Oriented Architecture

1 Scope

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

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

2 Terms and definitions

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

2.1 Definitions

2.1.1
actor
person or system component who interacts with the system as a whole and who provides stimulus which invoke actions

NOTE See ISO/IEC 16500-8:1999, 3.1.

2.1.2
architecture
fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution ISO/IEC/IEEE 42010:2011, 3.2). ISO/IEC 40210:2011

2.1.3
choreography
omposition whose elements interact in a non-directed fashion with each autonomous member knowing and following an observable predefined pattern of behavior for the entire (global) composition

NOTE See Bibliography Reference [21].

2.1.4
collaboration
omposition whose elements interact in a non-directed fashion, each according to their own plans and purposes without a predefined pattern of behavior

NOTE See Bibliography Reference [21].

2.1.5
composition
result of assembling a collection of things for a particular purpose

NOTE See Bibliography Reference [21].

2.1.6
effect
outcome of an interaction with a service

NOTE 1 If service contracts exist, they usually define effects. The effect is how a service, through the element that performs it, delivers value to its consumer.

NOTE 2 See Bibliography Reference [21].

2.1.7
element
unit that is indivisible at a given level of abstraction,and has a clearly defined boundary

NOTE 1 An Element can be any type of entity.

NOTE 2 See Bibliography Reference [21].

2.1.8
entity
individual in a service system with an identity which can act as a service provider or consumer

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

2.1.9
event
something that occurs,to which an element may choose to respond

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

2.1.10
execution context
set of technical and business elements that form a path between those with needs and those with capabilities and that permit service providers and consumers to interact

NOTE 1 The execution context of a service interaction is the set of infrastructure elements, process entities, policy assertions and agreements that are identified as part of an instantiated service interaction, and thus forms a path between those with needs and those with capabilities.

NOTE 2 See Bibliography Reference [19].

2.1.11
human actor
person or an organizational entity

NOTE 1 In principle, this classification is not exhaustive.

NOTE 2 See Bibliography Reference [21].

2.1.12
human tasks
tasks which are done by people or organizations, specifically instances of Human Actor

2.1.13
information Type
type of information given or received in a service interface

2.1.14
orchestration
composition for which there is one particular element used by the composition that oversees and directs the other elements

NOTE 1 The element that directs an orchestration by definition is different than the orchestration (Composition instance) itself.

NOTE 2 See Bibliography Reference [21].

2.1.15
process
composition whose elements are composed into a sequence or flow of activities and interactions with the objective of carrying out certain work

NOTE 1 A process may also be a collaboration, choreography, or orchestration.

NOTE 2 See Bibliography Reference [21].

2.1.16
REST
architectural style for distributed hypermedia systems. REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems

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

2.1.17
service
logical representation of a set of repeatable activities that has specified outcomes, is self-contained, may be composed of other services, and is a “black box” to consumers of the service

NOTE 1 See Bibliography Reference [21].

NOTE 2 The word “activity” in the ‘Service’ definition above is used in the general English language sense of the word, not in the process-specific sense of that same word (i.e., activities are not necessarily process activities).

2.1.18
service broker
implements service intermediaries that provide unified service registration and publishing

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

2.1.19
service bus
intermediary IT infrastructure that supports service access and consumption, event-driven message routing among services

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

2.1.20
service catalogue
service registry
service repository
component that supports publication, registration, search, and retrieval of metadata and artifacts for services

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

2.1.21
service choreography
composition whose elements are services that interact in a non-directed fashion with each autonomous member knowing and following an observable predefined pattern of behavior for the entire (global) composition

NOTE See Bibliography Reference [21].

2.1.22
service collaboration
composition whose elements are services that interact in a non-directed fashion, each according to their own plans and purposes without a predefined pattern of behavior

NOTE See Bibliography Reference [21].

2.1.23
service component
element that implements services

2.1.24
service composition
service assembly
result of assembling a collection of services to achieve a particular purpose

NOTE A composition can support different composition patterns: such as: collaboration, choreography, orchestration.

NOTE See Bibliography Reference [21].

2.1.25
service consumer
entity that uses services

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

2.1.26
service contract
terms, conditions, and interaction rules that interacting consumers and providers must agree to (directly or indirectly)

NOTE 1 A service contract is binding on all participants in the interaction, including the service itself and the element that provides it for the particular interaction in question.

NOTE 2 See Bibliography Reference [21].

2.1.27
service description
information needed in order to use, or consider using, a service

NOTE 1 The service description usually includes the service interfaces, contracts, and policies

NOTE 2 See Bibliography Reference [19].

2.1.28
service deployment
process that makes implementations of services deployed and able to actually run in a specific hardware and software environment

NOTE 1 Service deployment includes static deployment and dynamic deployment. Static deployment means that the call relations between the services is defined before runtime. Dynamic deployment means that when the application system is running, it needs to determine the call relations by dynamic routing.

NOTE 2 In terms of a single service, a service after deployment can be actually called by end users, other IT systems or services. In terms of multiple service-based processes, after deployment, they can form a complete application system to provide appropriate IT support for users.

2.1.29
service development
service implementation
technical development and physical implementation of the service that is part of a service lifecycle

2.1.30
service discovery
process that service consumers use to search and retrieve desired services according to their specific functional or non-functional requirements

2.1.31
service governance
strategy and control mechanism definition on service lifecycle, which includes establishment of chains of responsibility, ensures its compliance with policies by providing appropriate processes and measurements, addressing service modifications, version updates, notice of termination, decomposition subdivision, agency capacity, decomposition capacity, ability to meet individual demands

2.1.32
service interaction
activity involved in making using of a capability offered, usually across an ownership boundary, in order to achieve a particular desired real-world effect

NOTE See Bibliography Reference [19].

2.1.33
service interface
way in which other elements can interact and exchange information with a service as the outcome of a request in the definition of a service

NOTE See Bibliography Reference [21].

2.1.34
service interoperability
ability of providers and consumers to communicate, invoke services and exchange information at both the syntactic and semantic level

2.1.35
Service Level Agreement (SLA)
service contract that defines the interaction and measureable conditions of interaction between a service provider and a service consumer

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

2.1.36
service lifecycle
set of phases for a service throughout its life, from identification to instantiation and retirement

2.1.37
service management
monitoring, controlling, maintaining, and operating services

2.1.38
service modeling
service oriented analysis process of finding and modelling a series of service candidates for functions or actions which can be defined independently or by decomposing business processes

2.1.39
service monitor
monitoring and controlling operational state and performance of service

2.1.40
service orchestration
composition of services for which there is one particular element used by the composition that oversees and directs the other elements

NOTE 1 the element that directs an orchestration by definition is different than the orchestration (Composition instance) itself.

NOTE 2 See Bibliography Reference [21].

2.1.41
service orientation
approach to designing systems in terms of services and service-based development

2.1.42
service oriented analysis
preparatory information gathering steps that are completed in support of a service modeling sub-process that results in the creation of a set of service candidates

NOTE Service Oriented Analysis is the first phase in the cycle, though the service-oriented analysis process might be carried out iteratively, once for each business process. It provides guidance to the following process.

2.1.43
service oriented architecture
architectural style that supports service orientation and is a paradigm for building business solutions using IT

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

NOTE 2 See Bibliography Reference [21].

2.1.44
SOA governance
extension of IT governance specifically focused on management strategies and mechanisms for the end users' specific SOA solution

NOTE 1 It manages the entire SOA lifecycle by setting out personnel, roles, management procedures and decision-making. SOA governance needs to adopt the appropriate methodology and best practices. SOA governance usually requires tools for assistance to customize and manage the governance strategy according to the needs.

NOTE 2 While management means the specific process for governance and control to execute the policies, governance looks at assigning the rights to make decisions, and deciding what measures to use and what policies to follow to make those decisions.

2.1.45
SOA implementation
process methods and techniques used to develop SOA based solutions

2.1.46
SOA lifecycle
process for engineering SOA-based solutions, including analysis, design, implementation, deployment, test and management

2.1.47
SOA management
measurement, monitoring, and configuration of the entire lifecycle of a SOA solution

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

2.1.48
SOA maturity
quantitative description of the level of SOA application adoption within an IT architecture in an organization

2.1.49
SOA maturity model
framework and method to evaluate SOA maturity

2.1.50
service policy
statement that an entity may intend to follow or may intend that another entity should follow

NOTE See Bibliography Reference [21].

2.1.51
service provider
entity providing services

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

2.1.52
service publishing
publishing information for registered services

2.1.53
SOA resource
elements that provide the IT resources used by services

2.1.54
SOA solution
solutions implemented by applying SOA concepts, methods, and techniques

2.1.55
SOAP
stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information

NOTe See SOAP - SOAP Version 1.2 Part 0: Primer http://www.w3.org/TR/soap12-part0/

2.1.56
task
atomic action which accomplishes a defined result

NOTE See Bibliography Reference [21].

2.1.57
Web Services
software system designed to support interoperable machine-to-machine interaction over a network

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

2.2 Acronyms

ABB - Architectural Building Block

BMM – Business Motivation Model (see OMG)

BPMN – Business Process Management Notation

IT – Information Technology

EA – Enterprise Architecture

RA – Reference Architecture

SLA – Service Level Agreement

SOA - Service Oriented Architecture

SOSE – Service Oriented Software Engineering

SQL – Structured Query Language

WSDL – Web Services Description Language

WSRP – Web Services Remote Portlet

KPI – Key Performance Indicator

3 SOA Principles and Concepts

3.1 Introduction to SOA

Service Oriented Architecture (abbreviated SOA) is an architectural style that supports service orientation and is a paradigm for business and IT (see 3.1.40). This architectural style is for designing systems in terms of services available at an interface and the outcomes of services. A service is a logical representation of a repeatable business activity that has specified outcomes, is self-contained, may be composed of other services and is a “black box” to consumers of the service. (see 3.1.14).

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

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

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

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

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

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

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

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

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

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

However, a single set of SOA technical principles, specific norms, and standards have not been established for the world-wide market. Existing products and solutions have used various standards, methods and technologies. As a result, there is confusion about the effectiveness of SOA. To improve standardization and quality of solutions, as well as promote effective large-scale adoption of SOA, it is necessary to establish a unified set of general technical principles for SOA.

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

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

3.2 Concepts

3.2.1 Roles

Providers

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

- Operationally - the provider is responsible for responding to the exchange of messages with the consumer as well as producing the promised effect of invoking the service. Assuming the operational responsibility for providing a service implies the following across the lifecycle of said service:

    - Service Creation: Creating a service implementation that can provide the service in question
    - Providing Services: Providing the implemented service for use by others

- Publish Service: Publishing service descriptions (its interface and access information) to the service registry
- Hosting Services: providing a service container to support the runtime service interactions
- Governance: setting up business rules and policies for lifecycle management of the service

- Contractually - the provider is the entity that participates in the service contracts with the service consumer and is legally obligated to providing the service. Assuming the contractual obligation for providing a service implies the following across the lifecycle of said service:

  - Categorize Service: Deciding what category the service should be listed in for a given service registry
  - Service Pricing: Deciding how to price the services, or how/whether to exploit them for other value
  - Publish Service: Publishing the availability of the service as well as its promised effect
  - Defining Service Agreements: Decide what sort of formal agreements are required to use the service
  - Defining Service Contracts: Setting and abiding by the contracts.

  Governance: setting up business rules and policies for the service offerings

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

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

Consumers

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

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

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

Contractually - the consumer is the entity that participates in the service contracts with the service provider. Some of the responsibilities include:

- Contracts: setting and abiding by the contracts.
- Payments: Paying for services
- SLAs: Ensuring that services adhere to the service level agreements
- Governance: ensuring business requirements are met by the usage of the service

3.2.2 Services

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

To a consumer, a service is a black box, in other words, if two services have the same service contract and when given the same inputs will produce the same effects, they are equivalent to the consumer and should be able to be used interchangeably. To a provider, a service is a means of exposing capabilities and the

implementation determines equivalency. Therefore, two services that have the same inputs and produce the same effects but use different mechanisms are not equivalent.

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

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

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

- Pick the service to interact with (this statement is agnostic as to whether this is done dynamically at runtime or statically at design and/or construct time)
- Pick an element that performs that service (in a typical SOA environment, this is most often done "inside" an Enterprise Service Bus (ESB))
- Interact with the chosen element (that performs the chosen) service (often also facilitated by an ESB)

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

3.2.3 Semantics

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

3.2.4 Compositions and Processes

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

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

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

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

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

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

Services can represent or be implemented as a composition.

There are two subclasses of compositions that are especially important for SOA, Service Compositions and Processes.

Patterns of composition

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

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

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

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

Service compositions

Service compositions provide (in the operational sense) higher level services that are only composed of other services, yet can still exhibit the 3 composition patterns.

Processes

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

Processes can be composed of things like human actors, tasks, services, and processes. In addition, processes can exhibit any of the composition patterns.

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

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

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

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

Business Process

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

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

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

3.2.5 Service Registration and Discovery

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

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

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

During runtime, usage of a service by a consumer involves two steps – service discovery and location, and service invocation. The consumers of services interact with the Service Registry (needed to support Governance as well) to find the service. The Service Container invokes the Service Component to perform a service. Thus the functionality of the service and the physical service is the Service Component, while the role of the Services Layer is to act as the translation between the consumer and the Service Component.

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

The figure below show these steps.

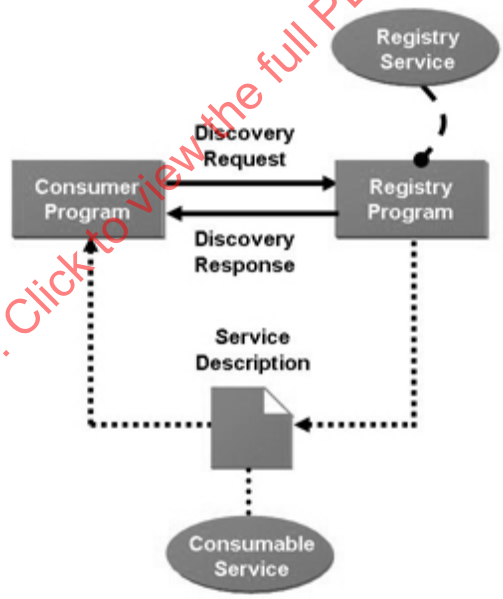

Figure 1 — Model for Service Discovery

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

Service Repository

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

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

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

Service Registry

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

For management of the service descriptions that it maintains

For enquiries and responses

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

3.2.6 Service Description, Interfaces, Contracts and Policies

Service descriptions can contain service interfaces, contracts and policies. Therefore we will define these before defining service description.

Service Interfaces

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

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

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

Contracts

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

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

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

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

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

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

Policy

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

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

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

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

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

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

Service Description

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

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

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

3.2.7 Service Lifecycle

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

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

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

3.2.8 SOA Lifecycle

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

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

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

- **Model –** Modeling is the process of capturing business design from an understanding of business requirements and objectives and translating that into a specification of business processes, goals and assumptions – creating an encoded *model* of the business, along with key performance indicators.

- **Assemble –** Assemble uses the business design to *assemble* the information system artifacts that will implement the business design. The business design is converted into a set of business process definitions and activities deriving the required services and from the activity definitions.

- **Deploy –** Deploy includes a combination of creating the hosting environment for applications and the actual deployment of those applications. This includes resolving the application's resource dependencies, operational conditions, capacity requirements, and integrity and access constraints. In

addition, the techniques you will employ for ensuring availability, reliability, integrity, efficiency, and service ability should be considered.

- **Manage –** Manage considers how to maintain the operational environment and the policies expressed in the assembly of the SOA applications deployed to that environment. This includes monitoring performance of service requests and timeliness of service responses; maintaining problem logs to detect failures in various system components; detecting and localizing those failures; routing work around them; recovering work affected by those failures; correcting problems; and restoring the operational state of the system.

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

3.2.9 Tasks and Activities

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

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

3.3 Architectural Principles

3.3.1 Architectural Principles defined

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

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

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

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

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

3.3.2 Interoperable – syntactic, semantic

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

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

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

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

Facilitated by:

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

Benefits are:

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

3.3.3 Described

The service description, as described in clause 4.2.6, is metadata that may include details on the, service contract, service interface and policies for both consumers and providers. The principle of being described is

achieved when the service consumer, provider, or developer can find, access, interpret, and process a description of a service.

Facilitated by:

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

Benefits are:

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

3.3.4 Reusable

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

Facilitated by:

- Service modeling and analysis – modeling the services to be developed to support the SOA solution, looking for common tasks and fine grained services in the solution
- Using the right granularity – finer grained 'utility' services may be reusable in service compositions
- Using autonomous definitions
- Adhering to principles for loosely coupled services
- Developing a well described service so that the service can be found, evaluated for appropriateness and reused.
- Making services discoverable
- Late binding of services, obtaining addresses for current service endpoint at runtime so that other services can be substituted at runtime and without redeveloping the service implementation
- Governance of the business with processes in place to identify redevelopment and require reuse of existing services

- Understanding how the service will be funded for development if it used across organizations
- Having the appropriate software development and maintenance lifecycle to support reuse of services in multiple solutions (i.e. impact when the implementation is updated)
- *Enabling management of services to be policy driven*
- *Implementing services with configuration isolated into policy*

Benefits are:

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

3.3.5 Discoverable

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

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

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

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

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

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

Facilitated by:

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

Benefits are:

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

3.3.6 Composable

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

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

Facilitated by:

- Adhering to the principle of reuse
- Developing autonomous services

Benefits are:

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

3.3.7 Self-Contained

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

Facilitated by:

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

Benefits are:

- Increases reusability
- Increased Composability

3.3.8 Loosely coupled

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

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

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

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

Loose Coupling facilitated a number of ways:

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

The benefits of loose coupling are:

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

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

3.3.9 Manageable

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

Facilitated by:

- Defining and monitoring key performance metrics for availability, reliability, performance and governance
- Providing interfaces for management capabilities along with interfaces for functional capabilities
- Enabling management of services to be policy driven
- Isolating configuration into policy
- Enabling configuration of services at design and runtime
- Enabling monitoring of adhering to contracts
- Development of governance policies and processes
- Enabling control over the lifecycle of the service, (enable, disable)

Benefits are:

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

3.4 Cross Cutting Aspects

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

Critical success factors for a deployable SOA architecture are:

3.4.1 Integration

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

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

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

Cross Domain interaction:

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

The Figure 2 describes shows a number of solution domains, each implemented with its own technology, where the interactions between them need to be managed as a whole.

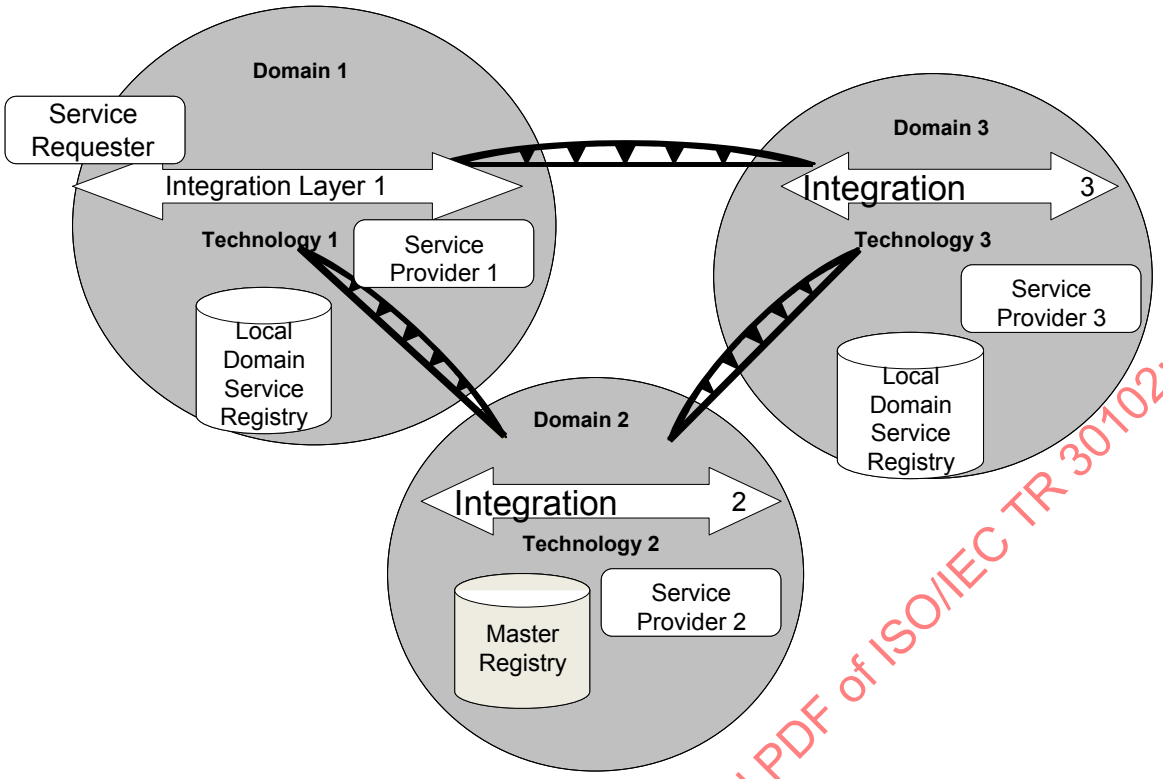

Figure 2 — Cross Domain Interaction

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

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

Service Integration

Service integration as it provides the capability to mediate, transform, route and transport service requests from the service requester to the correct service implementation. This layer may include an intelligent routing, protocol switching, and interface transformation mechanisms as is often seen in an Enterprise Service Bus (ESB). That integration layer enables the loose coupling between the request and the concrete provider by matchmaking the Service Request and Service Implementation.

This loose coupling provided by the integration layer is not only a technical loose coupling addressing protocols, locations or platforms, but can also be a business semantic loose coupling performing required adaptations between service requester and provider.

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

There are 3 aspects of integration that need to be considered are transportation, transformation and mediation.

Transport

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

Transformation

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

Mediation

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

3.4.2 Management and Security

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

Transaction

Transactions coordinate the service's actions on resources that can belong to distributed components when these components are required to reach a consistent agreement on the outcome of the service's interactions.

A transaction manager usually coordinates the outcome while hiding the specifics of each coordinated service component and the technology of the service implementation. With that approach a common and standardized transactional semantic can be used on the services.

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

Atomic transaction

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

Business activity transactions

Business activity transactions occur when building choreographies that require consistent agreement on the outcome of coordinated long-running distributed services interactions where locks cannot be maintained on modified resources for the time scale of the choreography. Because of the usual stateless aspects of services the changes that occurred on the resources managed by a service component cannot be rolled back after the

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

Availability

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

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

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

Service Reliability

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

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

Four reliability qualities may be provided by SOA systems:

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

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

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

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

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

SOA Security

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

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

The threats that SOA security needs to protect from are the following.

- Destruction (an attack on availability): Destruction of information and/or resources and/or components accessed through services or related to service and service lifecycle.
- Corruption (an attack on integrity): Unauthorized tampering with an asset accessed through services or related to service and service lifecycle.
- Removal (an attack on availability): Theft, removal or loss of information and/or other resources affecting services
- Disclosure (an attack on confidentiality): Unauthorized access to an asset or a service
- Interruption (an attack on availability): Interruption of services. Service becomes unavailable or unusable

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

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

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

6. Data Integrity: Ensure that data is received as sent or retrieved as stored in all of the interactions between service components, whether they are for SOA management or lifecycle purposes or for interactions between service consumers and providers. Examples: MD5, digital signature with XML-Signature, anti-virus software. The XML signature should only be processed at both end to avoid opening a breach in the various nodes that constitute an SOA architecture between the service consumer and provider.

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

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

SOA Security Governance

A specific domain of governance addresses the SOA security.

- SOA Security functions: The 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: The SOA security governance defines and implements the components that administer security policies, the security policies distribution & transformation to the appropriate SOA components and elements, the security policy decision & enforcement components and finally the components that monitor and report on security policies use and conformance.

- SOA Security processes: finally the SOA security governance defines the IT and business processes for risk and compliance, trust management, identity & access control lifecycle, data protection & disclosure control, and finally security for all components in a service oriented architecture such as registries, ESB etc. In a multi entity environment these process may include the definition and exchange of certificates or public keys between trusted actors before enabling other actors in the entities to interact.

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

SOA Management

The same kinds of management that apply to businesses today are important for managing services and SOA solutions and may need extensions to handle the service oriented nature and the cross domain boundaries of many SOA solutions. While Service Management is focused on the management of services, SOA management is has a broader scope and manages the entire SOA solution, or set of SOA solutions. Common types of management used for SOA solutions are:

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

Application and SOA Monitoring and Management: Provides monitoring and management of software services and application, this includes ability to capture metrics and to monitor and manage application and solution status.

- **Business Activity Monitoring and Management:** Provides monitoring and management of business activities and business processes. It provides ability to analyze this event information, both in real-time / near real-time, as well as stored (warehoused) events and to review and assess business activities in the form of event information and determines responses or issues alerts/ notifications.

- **Security Management**: Manages and monitors security and secure solutions. This provide ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, address how software, systems and services are developed and maintained throughout the software lifecycle, maintain the security status through proactive changes reacting to identified vulnerabilities and new threats, enable the IT organization to manage IT related risks and compliance, and provide the automation basis for security management.
- **Event Management:** Provides the ability to manage events and enables event processing, logging, and auditing.
- **Configuration and Change Management:** This category of capabilities provides ability to change solution and service configuration and descriptions.
- **Policy Monitoring and Enforcement:** Provides mechanism to monitor and enforce of policies and business rules for the SOA solution and services. This includes finding and accessing policies, evaluating and enforcing policies at check points. Policy enforcement includes enforcement when metrics are captured, signalled and recorded. Enforcement must also send compliance status or metrics, as well as notification and log of non-compliance
- **Lifecycle management:** Provides a mechanism to deploy, start/enable, stop/disable, and undeploy services and SOA solutions.

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

Management Services

Management services represents the set of management tools used to monitor service flows, the health of the underlying system, the utilization of resources, the identification of outages and bottlenecks, the attainment of service goals, the enforcement of administrative policies, and recovery from failures. Since we can capture the business design as a model, and use that to assemble the application services that implement that design, we can capture a correlation between the business and the IT system. 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. (See[.20])

Service Metadata Management

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

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

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

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

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

3.4.3 SOA Governance

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

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

This clause is sourced from The Open Group SOA Governance Framework Technical Standard (see [22]).

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

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

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

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

The SOA Governance Framework can enable enterprises to define and deploy their own focused and customized SOA governance regimens, based on enterprise-specific factors such as business model, SOA maturity, and company size. It describes what decisions must be made, who should make them, how they are made and monitored, and what organization and tools will support them. It uses an incremental approach so that enterprises can continue to meet their current SOA demands while fulfilling their long-term aspirations and goals for SOA.

In order to specify the changes necessary to accommodate SOA in an existing governance regime, these governance activities must be mapped and integrated to the activities being utilized in the existing regime. While no two governance regimens are alike, governance standards can define best practices and provide a starting point for customization to the SOA solution.The SOA Governance Reference Model can be defined and used to develop an enterprise's customized SOA governance regimen. It should include governance activities that are impacted by SOA:

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

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

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

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

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

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

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

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

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

- Plan: identifies the requirements for SOA governance, reviews the existing governance regimen, and defines the vision, scope, and strategy for the changes to be made.
- Define: creates a tailored SOA governance regimen from the SOA Governance Reference Model, using the results of the Plan phase. It then analyzes the differences between this regimen and the existing governance regimen to create transition plans.

- Implement: realizes the tailored SOA governance regimen that was created in the Define phase to produce an SOA governance regimen for the enterprise; essentially it executes the transition plans from the define phase.
- Monitor: monitors the effectiveness of the operation of the SOA governance regimen that was produced in the Implement phase and whether it is meeting its intended purpose. This generates requirements for change that can be addressed in a new cycle of the SOA Governance Vitality Method.

Facilitated by:

- Repository
- Communications
- Centers of Excellence, Governance Boards

Benefits of:

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

4 SOA Technical Framework

4.1 Introduction to the SOA Technical Framework

The SOA Technical Framework consists of two complementary parts, first a reference architecture for SOA solutions, and second, a set of common service types or categories. Both of these will help architects to develop architectures and models for their SOA solutions, the business applications which have been developed using the service oriented architectural style. This includes all of the design and runtime components, physical and logical, to develop, deploy, and operate this business application. The complementary parts are enumerated as follows:

1. The SOA Reference Architecture enumerates the fundamental elements of a SOA solution or enterprise architecture standard for solutions and provides the architectural foundation for the solution by specifying the capabilities and architectural building blocks required to implement those capabilities.

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

Each layer supports, and is defined in terms of: requirements, logical, and physical aspects. The requirements aspect reflects what the layer enables and includes all of its capabilities; the logical aspect includes all the architectural building blocks (ABBs), design decisions, options, Key Performance Indicators (KPIs), etc; while the physical aspect of each layer includes the realization of each logical aspect using a particular chosen technology platform, standards and products, that are determined by taking into consideration the different options and selecting one along with the documentation of architectural decisions that led to the decision and that are necessary to be made to realize and construct the architecture. The actual realization of the architecture with be through a set of products or platforms that will be left open to the implementer of the standard.

This clause documents a high level description of each of the layers. The detailed descriptions of the requirements and capabilities, logical architectural building blocks, and physical mapping are not in the scope of this technical report.

2. The Common Service Categories clause documents a range of common domain specific (functional) and non domain specific (supporting) categories of services that can be used as a checklist to helps architect understand and decide which of these types of services are appropriate for their solution.

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

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

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

Some of the diagrams and text in clause 5 are derived from contributions:

- The Open Group SOA Reference Architecture Technical Standard (see [20] for more information)
- China's Technical Reference Architecture for SOA Solutions (see Annex D)
- SOA Related Function - Japanese Technical Reference Model (TRM) for the Government Procurement of Information Systems (see Annex F).

4.2 Reference Architecture for SOA Solutions

This clause presents a reference architecture for SOA solutions. It provides a high level abstraction of a SOA partitioned and factored into layers, each of which provides a set of capabilities required to enable the working of a SOA. Each layer addresses a specific subset of characteristics and responsibilities that relate to unique value propositions within a SOA.

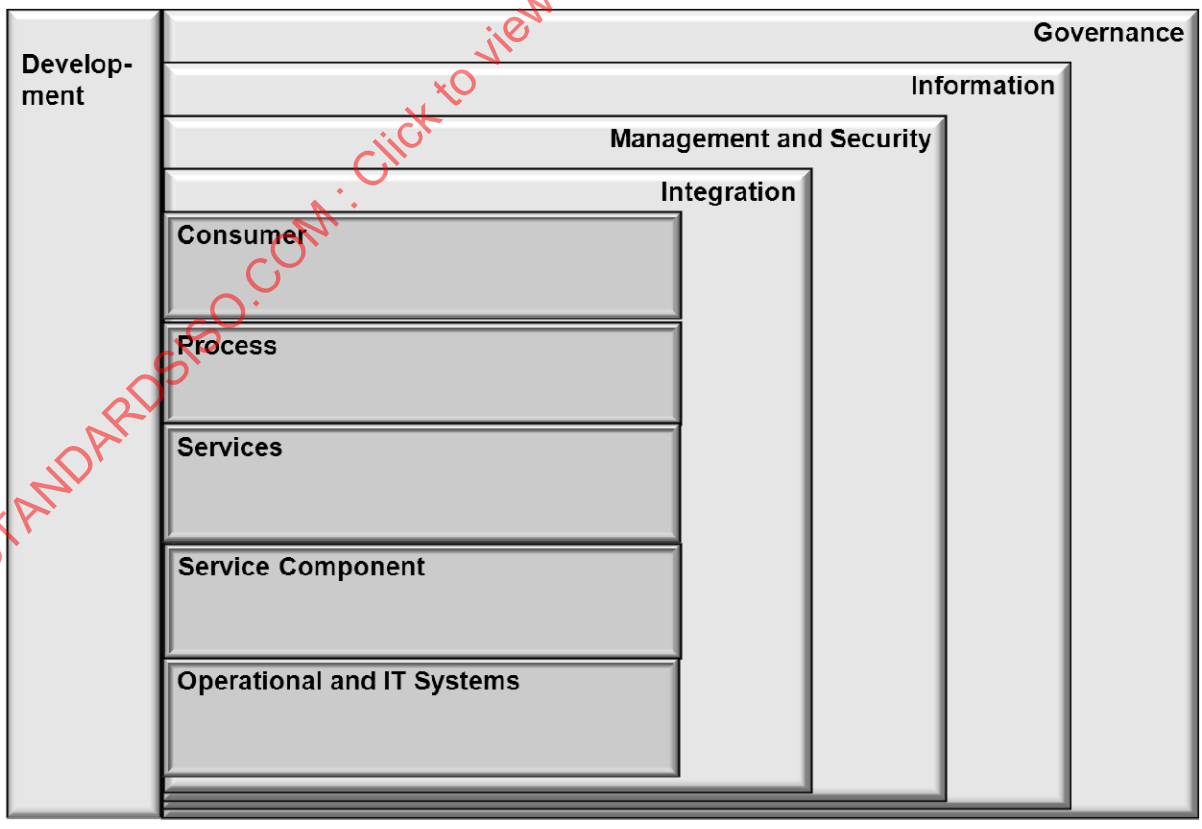

Figure 3 — Reference Architecture for SOA Solutions

Figure 3 illustrates the multiple separations of aspect in the ten layers of this reference architecture.

Figure 3 depicts a SOA as a set of logical layers. Note that one layer does not solely depend upon the layer below it and is thus named a partially-layered architecture: a consumer can access the Process Layer or the Service Layer directly, but not beyond the constraints of a SOA architectural style. For example, a given SOA solution may exclude a Process Layer and have the Consumer Interface Layer interacting directly with the Service Layer. Such a solution would not benefit from the business value proposition associated with the Process Layer however that value could be achieved at a later stage by adding the layer. In this sense the SOA Reference Architecture represents SOA with a partially layered architecture. The degree to which a given organization realizes the full SOA Reference Architecture will differ according to the level of SOA maturity they exhibit, and the underlying requirements of the organization.

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

The SOA Reference Architecture does not assume that the provider and the consumer are in one organization, and supports both SOA within the enterprise as well as across multiple enterprises in the industry ecosystem. The need for both intra-and inter-enterprise SOA is important, since SOA a foundation of Cloud Computing.

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

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

4.2.1 Operational and IT Systems Layer

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

- All infrastructures to run the SOA and its components.
- All operational and runtime hosting of components, both physical and infrastructural.
- All deployment time components.
- All resources to support services, data, and application system.

- All assets required to support the functionality of the service in the SOA, including custom or packaged application assets, new services, services created through composition or orchestration, infrastructure services, etc.

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

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

- Existing monolithic custom applications
- Existing transaction processing systems
- Existing databases
- Existing package applications and solutions including ERP and CRM packages
- Legacy applications and systems
- Web services systems
- IT Infrastructure
- Enterprise Application Integration (EAI) systems
- Service Resources providing access to existing business elements or 3rd party
- Data Resources providing physical storage data in the business solution
- Application System Resources providing specific business functions
- Existing services

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

4.2.2 Service Components Layer

The Service Component layer contains components which provide the implementation or “realization” for a service, or operation on a service; hence the name Service Component. The layer contains the functional and technical components that facilitate a Service Component to realize one or more services. Service components reflect the definition of the service they represent, both in its functionality and its management and quality of service interactions. They “bind” the service contract to the implementation of the service in the operational and IT systems layer. Service components are hosted in containers which support a service specification.

The service component layer manifests the IT conformance with each service contract defined in the services layer; it guarantees the alignment of IT implementation with service description.

Each Service component:

- Realizes one or more services

- Provides an enforcement point for “faithful” service realization (ensures quality of service and service level agreements)
- Enables IT flexibility by strengthening the decoupling in the system by hiding volatile implementation details from service consumers.
- Offers a façade behind which IT is free to do what they want/need to do
- Generally contains business specific logic with no reference to integration logic

The service component layer enables IT flexibility through encapsulation and by enabling loose coupling. The separate of concerns such that the consumer assumes that the realization of the service is faithful to its published description (service compliance) and the providers’ ensures that such compliance is achieved. The details of the realization are of no consequence to the consumer. Subsequently, the Provider organization may decide to replace one component with another with the result that there is no impact on any consumers of the *Service*.

4.2.3 Services Layer

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

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

The description provides consumers with sufficient detail to invoke the business functions exposed by a provider of the service; ideally this may be done in a platform independent manner. Service descriptions may include:

- Descriptions of the abstract functionality offered by the service similar to the abstract stage of a WSDL description (see [27]). Note that the use of WSDL is illustrative and the description can be done in any language supporting description of the functionality.
- Policy documents.
- SOA management descriptions.
- Attachments that categorize or show service dependencies.

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

This layer contains the contracts (service descriptions) that bind the provider and consumer. Services are offered by service providers and are consumed by service consumers (service requestors). Service Components or existing enterprise applications (legacy systems, packaged applications, etc.) are responsible for the actual implementation *aka* realization of a service. At runtime, this implementation will reside in a container within the Operational Systems Layer, which is responsible for the runtime.

The Services Layer supports:

- Functional capabilities aka services that enables business capabilities that businesses perform to achieve a business outcome.
- Supporting capabilities to define and specify the “services” in terms of service interface/contract/description, message specification and policy descriptions.
- Supporting capabilities to enable the runtime execution of service and the support of service virtualization.

4.2.4 Process Layer

The Process Layer covers the process representation, composition methods, and building blocks for aggregating loosely coupled services as a sequencing process aligned with business goals. Data flow and

control flow are used to enable interactions between services and business processes. The interaction may exist within an enterprise or across multiple enterprises.

Business capabilities are realized by business processes, or collections of business processes. These business processes include service orchestrations and compositions, and the ability to insert “human intervention” and support long-lived transactions. The evolution of service composition into flows, or choreographies of services bundled into a flow act together to establish an application. These applications support specific use cases and business processes

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

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

In summary, the process layer in the SOA Reference Architecture plays a central coordinating role in connecting business level requirements and IT-level solution components through collaboration with integration layer, security and management layer, as well as information architecture layer and services layer.

4.2.5 Consumer Interface Layer

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

The Consumer Interface Layer of the SOA Reference Architecture provides the capability to quickly create the front end of the business processes and composite applications to respond to changes in the marketplace. It enables channel independent access to those business processes supported by various application and platforms.

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

Thus, the Consumer Interface Layer provides the capabilities required to deliver IT functions and data to end-users and as the entry point for consumers into the SOA RA. These capabilities enable specific users to customize preferences, integrate with consumer channels, including rich clients, i.e. mashups and Ajax (see [33]) and act as a mechanism for the underlying SOA to expose its functionality. Standards such as WSRP (Web Services Remote Portlets see [28]) may leverage services at the application interface or presentation level. Its capabilities include the ability to quickly create the front end of the business processes and the composite applications to respond to changes in the market. It provides the point where in-bound consumer requests have security and other quality of service policies asserted to ensure that the request is secure and brought into the context of the SOA.

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

4.2.6 Integration Layer

The integration layer enables and provides the capability to mediate, which includes transformation, routing and protocol conversion to transport service requests from the service requester to the correct service provider. It supports the capabilities required for enabling SOA such as routing, protocol support and conversion, messaging/interaction style, support for heterogeneous environment, adapters, service interaction, service enablement, service virtualization, service messaging, message processing and transformation.

For more information on the challenges and concepts for integration of services in SOA Solutions, see clause 4.4.2.

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

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

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

The integration that occurs here is primarily the integration of service component, service, and process layers (the “functional” layers). For example, this is where binding (late or otherwise) of services occurs for process execution. This allows a service to be exposed consistently across multiple customer facing channels such as Web, IVR, Seibel client (used by Customer Service Rep), etc. The transformation of response to HTML (for Web), Voice XML (for IVR), XML string can be done via XSLT functionality supported through mediation service transformation capability in the Integration layer.

4.2.7 Management and Security Layer

The Management and Security layer supports non functional requirement (NFR) related issues as a primary feature/concern of SOA and provides a focal point for dealing with them in any given solution. It provides the means of ensuring that a SOA meets its requirements with respect to: 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).

Management and Security is especially important for SOA solutions to enable the loosely coupled solutions for scaling and the efficiency necessary to fulfill non-functional requirements. This layer maintains and ensures the “quality of service (QOS)” aspect, as discussed in clause 4.4.3, in computer systems through a combination of: increased virtualization / loose coupling, widespread use of XML, the composition of federated services, heterogeneous computing infrastructures, decentralized SLAs, the need to aggregate IT metrics to produce business metrics etc.

To enable both Management and Security, this layer:

- Provides solution management of various aspects, such as availability, reliability, security, and safety as well as Includes the mechanisms to support, track, monitor, and manage solution quality controls

- Provides the ability to monitor and enforce multitude of policies and corresponding business rules including business level policies, security policies, access privileges, data access policies etc.

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

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

- Supports the ability to monitor and manage both at the 1) business level in terms of key performance indicators (KPIs), events, and activities in the business processes 2) IT systems level for the security, health and well being of IT systems, services, applications, networks, storage, and compute servers.

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

The same kinds of management and monitoring that apply to businesses today 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. The capabilities supported by SOA solutions are:

1. **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.
2. **Application and SOA Monitoring and Management:** Provides monitoring and management of software services and application, this includes ability to capture metrics and to monitor and manage application and solution status.
3. **Business Activity Monitoring and Management:** Provides monitoring and management of business activities and business processes. It provides ability to analyze this event information, both in real-time / near real-time, as well as stored (warehoused) events and to review and assess business activities in the form of event information and determines responses or issues alerts/ notifications.
4. **Event Management:** Provides the ability to manage events and enables event processing, logging, and auditing.
5. **Configuration and Change Management:** This category of capabilities provides ability to change solution and service configuration and descriptions.
6. **Policy Monitoring and Enforcement:** Provides mechanism to monitor and enforce of policies and business rules for the SOA solution and services. This includes finding and accessing policies, evaluating and enforcing policies at check points. Policy enforcement includes enforcement when metrics are captured, signalled and recorded. Enforcement must also send compliance status or metrics, as well as notification and log of non-compliance
7. **Lifecycle management:** Provides a mechanism to deploy, start/enable, stop/disable, and undeploy services and SOA solutions.

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

The capabilities explicitly for security are:

Security Management: Manages and monitors security and secure solutions. This provide ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss, address how software, systems and services are developed and maintained throughout the software lifecycle, maintain the security status through proactive changes reacting

to identified vulnerabilities and new threats, enable the IT organization to manage IT related risks and compliance, and provide the automation and audit basis for security management.

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

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. Important areas of policy enforcement are security, messaging transportation, and infrastructure availability, and service availability. Responses (dispensations and appeals) to non compliance and exceptions are defined by the Governance Layer as well.

4.2.8 Information Layer

The Information Layer is responsible for manifesting a unified representation of the information aspect of an organization as provided by its IT services, applications, and systems enabling business needs and processes and aligned with the business vocabulary – glossary and terms.

The information layer includes information architecture, business analytics and intelligence, meta-data considerations and ensures the inclusion of key considerations pertaining to information architectures that can also be used as the basis for the creation of business analytics and business intelligence through data marts and data warehouses. This includes meta-data content that is stored in this layer. It also supports the ability for an information services capability, enabling a virtualized information data layer capability. This enables the SOA to support data consistency, and consistency in data quality.

The Information layer supports these capabilities:

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

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

4.2.9 Governance Layer

SOA Governance ensures that the services and SOA solutions within an organization are adhering to the defined policies, guidelines and standards that are defined as a function of the objectives, strategies and regulations applied in the organization and that the SOA solutions are providing the desired business value. SOA governance activities shall conform to Corporate, IT and Enterprise Architecture governance principles and standards. The Governance layer will be adapted to match and support the target SOA maturity level of the organization.

To understand concepts of SOA governance and security governance see clause 4.4.4.

The governance layer includes both SOA governance (governance of processes for policy definition and enforcement) as well as Service governance (service life-cycle). This covers the entire lifecycle of the service and SOA solutions as well as the portfolio management of both the services and SOA solutions managing all aspects of services and SOA solutions (e.g. SLA, capacity and performance, security and monitoring).

The goal of the SOA Governance layer is to ensure consistency of the Service and Solution portfolio and lifecycles processes. In this layer, the extensible and flexible SOA governance framework will ensure that business and IT continue to remain aligned, including:

- Service Level Agreements based on requirements for quality of service and key performance indicators (KPIs)
- Capacity and performance management policies
- Design time aspects such as Business Rules

As a part of the governance framework, the Governance regimen, i.e. the customized compliance, dispensation, and communication processes to govern the SOA lifecycle and portfolio management, requires capabilities to store and access governance artifacts, manage and enforce policy, monitor metrics, and manage the configuration of the solution and governance. Change control may be needed to support changes to the system.

To ensure ongoing business and IT alignment, the Governance Vitality method phases, Plan, Define, Implement, and Monitor should be executed continuously. All of these phases require the capability to store and access governance information; define policies with a policy manager, and possibly development and configuration management tools. In addition the monitor phase needs the ability to monitor metrics, manage and enforce policies, and use change control and configuration management tools to react to policy changes. In addition workflow can be used to implement the compliance processes.

This layer supports the following capabilities:

- ability to defines policies, compliance and exceptions characteristics
- ability to monitor the health of SOA services, solution and governance via the management and security layer
- ability to report on compliance, exceptions, service health, versions
- ability to provides a consolidation point for business rules

4.2.10 Development Layer

The development layer contains all of the components and products needed to develop and change implementations of SOA services and solution. The service implementations should include the development of or use of implementations in the operational systems and IT layer, service component layer, and cross cutting aspects layers. Service implementations should encapsulate the existing systems and resources such that late binding of services can be supported to promote loose coupling.

Development includes solution and service design, modeling, implementation and deployment. Operational and maintenance phases are the responsibility of the management and security layer.

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

The Service development layer supports the capabilty of:

- Providing development, configuration, debugging and testing environments for the construction of services.
- Service encapsulation of existing application systems or data resources.

- Reusing existing assets to develop services

4.3 Common Services Categories

4.3.1 Common Services Categories Overview

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

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

The figure 4 below shows a functional categorization scheme for services found in a typical enterprise. As mentioned above, this type scheme is for the services themselves, not for their implementations as their implementations will use other layers of the reference architecture. In this case, these categories are loose groups of services that support the same function where membership in a category is not mutually exclusive.

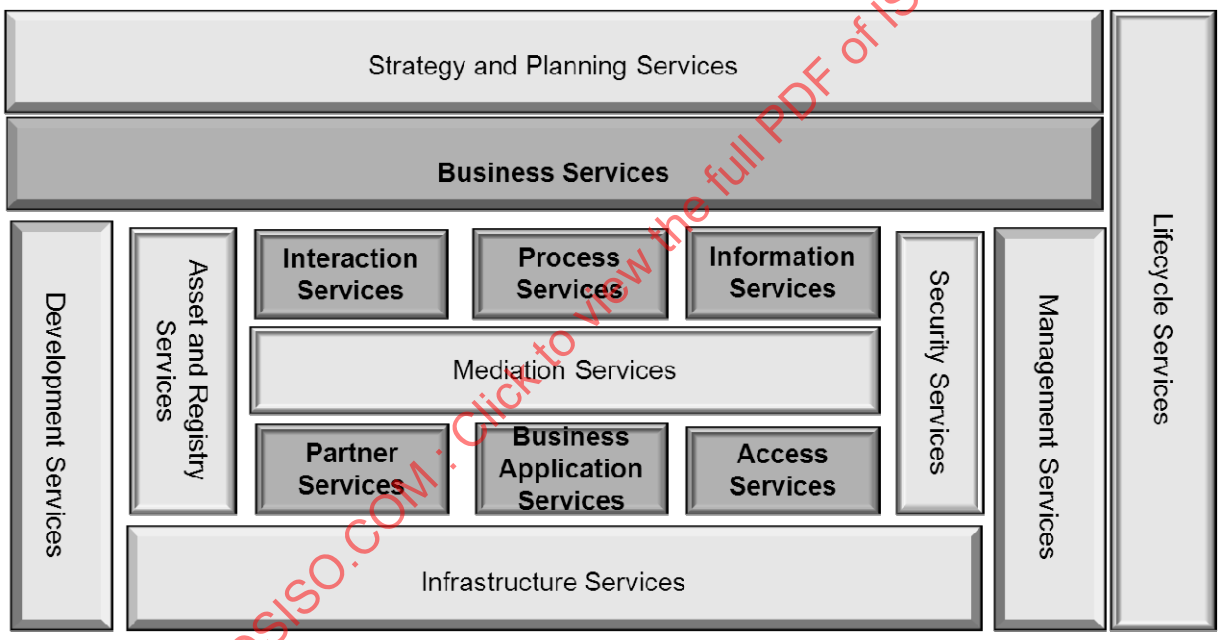

Figure 4 — Common Service Categories

The categories of services are broken down in Figure 4. Dark colored services such as interaction services, process services, information services, etc. are considered to be domain specific. Services that are domain specific service categories are solution specific and thus require implementations that are unique to the domain or solution being developed. Sometimes, domain specific services can be purchased, but generally they require extensive customization or extension.

The remaining services categories are considered to be domain independent. These domain independent categories include development services, management services, etc. Services of this category can be used directly in many different domains or solutions. In general domain independent services are used to plan, develop, support and manage the domain specific services in the solution. Often domain independent services can be purchased and used without extensive customization.

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

Services categories are:

4.3.2 Mediation Services

Mediation services are a category of services that assumes the responsibility for binding service consumers with service providers – they implement this responsibility by resolving their location automatically to achieve an optimal routing of requests across the network and meet the goals of the business. The presence of mediation services should be transparent to the consumer of functional services in the SOA solution. Mediations offered by mediation services leverage the connectivity and typically adds additional value by doing some useful activity in addition to the connectivity.

Implementations of mediation services perform message transformation, intelligent routing, augmented functionality (such as logging or auditing) to enable the interconnectivity of services. They encompass the mediations as well as the services that host them. Since the mediation services, like ESBs, are a transparent fabric interconnecting service consumers with service realizations/implementations, they hide any hosting of mediating logic, and the topology of services, consumers, and providers being mediated.

4.3.3 Interaction Services

Interaction services are a category of services that provides the presentation logic of the business design. These services are components that support the interaction between applications and end-users. Interactions with the external world are not limited to just interactions with humans, interaction logic orchestrates the interface to all kinds of devices and control systems, including vehicles, sensors, and RFID devices, Every external interaction projects a view of the information system that is tailored for the specific interaction fidelity, frequency of interaction, and presentation composition that best fits the needs of the end user or device.

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

4.3.4 Process Services

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

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

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

4.3.5 Information Services

Information services are a category of services that contain the data logic of business design. The service implementations that provide the data logic have three major responsibilities: to provide access to the persistent data of the business, to support data composition of the business, to provide their own sub-architecture for managing the flow of data across the organization.

- **Data Access:** The data access information service implementations can include query statements for retrieving information or referential integrity checks on the information manipulated by these service implementations. Information services for data access incorporate federation of multiple data sources.

- **Data Composition:** The data composition information service implementations compose information in a way that matches the composition of services in the business design. This is analogous to the kind of re-factoring that can occur with legacy applications to get them to fit better with the business design. In addition, it is common practice to implement these services to separate the database design from the application design to achieve the level of performance and scalability required in many enterprise computing environments.

- **Data Flow:** The data flow information service implementations manage the movement of information from one part of the enterprise to another. The movement of data is needed to satisfy its own data flow and lifecycle requirements. This may involve the use of Extract-Transform-Load (ETL) mechanisms to process and enrich data in bulk, batch processing activities involved in bulk transaction processing, and migrating data from master-data-of-record databases to information warehouses that can be used to perform post-processing and business intelligence, analytics, and content management functions – which in turn are made available to the business application as services.

4.3.6 Access Services

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

4.3.7 Security Services

Security services are a category of services that address the protection against threats across the vulnerable dimensions of an SOA. The protecting of interactions between service consumers and service providers is only one aspect of security services. They are also responsible for protecting all of the elements that contribute to the architecture.

Some of the threats that SOA security needs to protect from are the following.

- Destruction (an attack on availability): Destruction of information and/or resources and/or components accessed through services or related to service and service lifecycle.
- Corruption (an attack on integrity): Unauthorized tampering with an asset accessed through services or related to service and service lifecycle.
- Removal (an attack on availability): Theft, removal or loss of information and/or other resources affecting services
- Disclosure (an attack on confidentiality): Unauthorized access to an asset or a service
- Interruption (an attack on availability): Interruption of services. Service becomes unavailable or unusable

Security services for SOA solutions should include those that support:

- providing access control and authentication management of related resources.
- providing secure interaction services, including the security protection measures during the transmission process to prevent information tampering, leaking, etc.
- providing security for the information/data of the services. It should provide encryption and decryption, signature, data integrity verification and other services for information / data and other resources.
- providing auditing and logging services, including access to audit logs, history and tracking logs, and statistical services.

4.3.8 Partner Services

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

4.3.9 Lifecycle Service

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

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

4.3.10 Asset and Registry Services

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

4.3.11 Infrastructure Services

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

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

4.3.12 Management Services

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

4.3.13 Development Services

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

4.3.14 Strategy and Planning Services

Strategy and planning services are a category of services that supports creating vision, blueprint and transition plan for improving business outcomes. Specifically these are services that process the strategies of the business to create an implementation roadmap covering both business and IT. In other words these services support the long term evolution and effectiveness of an enterprise.

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

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

4.3.15 Business Application Services

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

4.3.16 Business Services

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

Business services should be have key performance indicators – business objectives and general metrics - defined and monitored in order to provide business executives, business analysts and other human experts information to ensure the SOA and the service is meeting business objectives.

4.3.17 Considering Implementations of Common Service Categories using Reference Architecture

As described in clause 5.2.3, all logical services that are in the common services categories are logically located in the services layer, just as all services are. However, some of the category names are common with the Reference Architecture names. This is because there is an affinity between the categories and certain layers of Reference Architecture for SOA Solutions based on the semantics and where the bulk of the implementation of a service is, as illustrated in the following diagram. The layer association is not exclusive, indeed implementations of these services, just like all other services, use or implement capabilities and building blocks from all of the cross cutting layers, the Service Component Layer and the Services Layer to fulfill their functional capabilities.

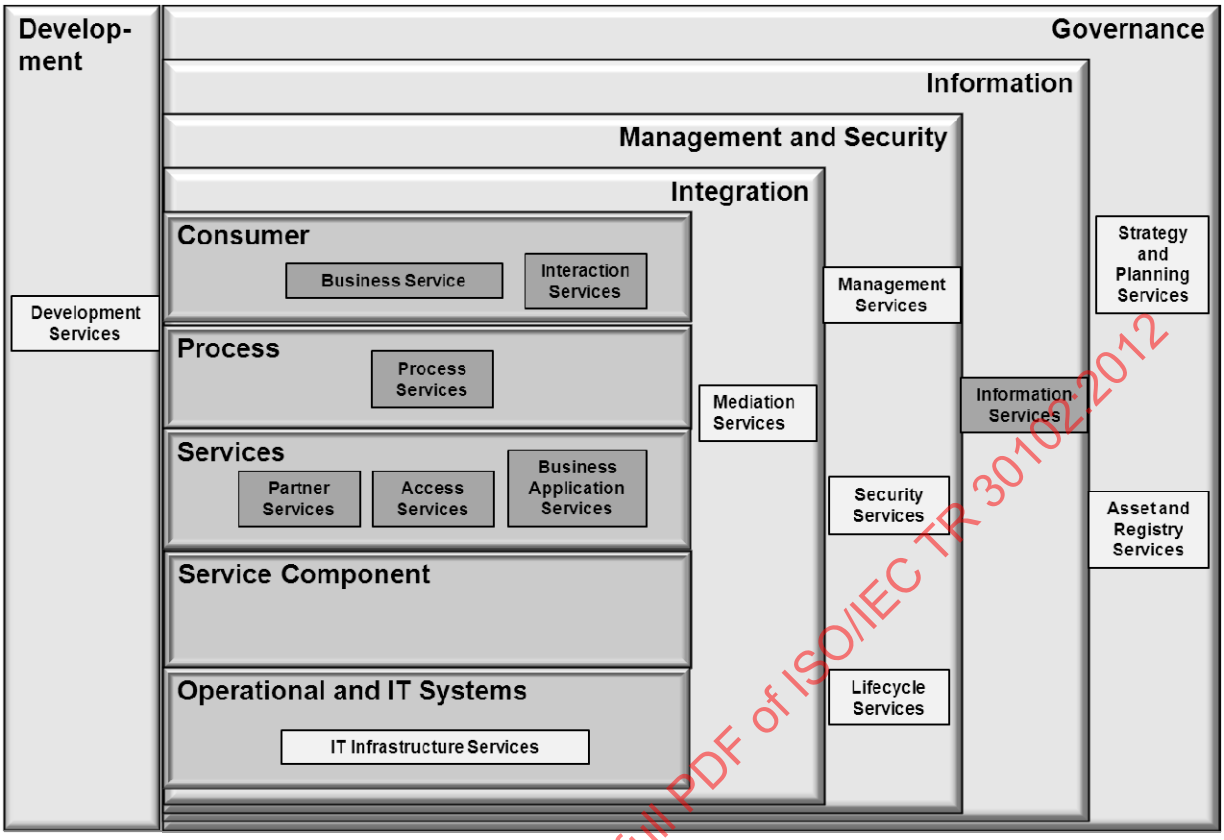

Figure 5 — Affinity between Service Categories and layers of RA for SOA Solutions

For example, IT Infrastructure services provide access to many resources in the Operational and IT Systems layer. Yet implementations of infrastructure services also use or implement capabilities and ABBs in the integration layer, Management and Security layer, Service component layer and Services layer.

Implementations of partner, access, and business application services in the service layer provide access to and encapsulation of interactions with third party systems, operational systems and business systems respectively. These service implementations are domain specific, aka, they offer functionality that is part of the semantics of the SOA solution and are generally implemented specifically for a particular SOA solution.

Implementations of process services implement and provide access to the process layer capabilities, yet are also consumers of partner, access and business application services.

For the consumer Interface layer, interaction services support the consumers by supporting interactions through different channels. Business services are services that consumers will interact with directly, yet their implementations are also consumers of other services like process services and business application services.

Mediation services offer many of the capabilities of the integration layer for binding, transforming, and transporting services.

Management services and Security services implementations offer access to the functionality and architectural building blocks in the Management and Security layer, like identification management and polling for metrics. Implementations of lifecycle services support tasks like deployment, configuration, change control, enabling services, and disabling services, yet in turn consume asset and registry services.

The governance layer provides business guidelines and policies to be implemented and enforced in the SOA solution. The Strategy and Planning services support (amongst other things) the setting of those guidelines

and policies. Asset and registry services are implemented to provide access to registry capabilities in the governance layer.

These categories of services can be kept in mind when developing your Service portfolio and your SOA solution portfolio. Use these as a checklist to ensure that you have considered all the relevant kinds of services and can make the right choices on fulfilling the development or purchase of services that are in scope.

Annex A
(informative)

The Open Group SOA Reference Architecture

The Open Group SOA Reference Architecture (see [20]) is intended to support the understanding, design, and implementation of common system, industry, enterprise, and solution architectures leveraging the principles of an SOA.

This SOA reference architecture provides the basis, or blueprint, for an enterprise architecture so that the enterprise architect can use that template or blueprint as a standard that will be instantiated during each individual project or solution that is being developed. This will be performed within the organization where the SOA reference architecture will be instantiated.

This SOA reference architecture is designed to support different kinds of scenarios including those involving consumer organizations, vendors, other standard bodies, and other Open Group projects. Specifically The Open Group SOA Reference Architecture:

Assists and guides consumer organizations designing and implementing an SOA by providing a concrete basis for evaluating and making architectural and design decisions

Supports and provides a vehicle for vendors using this SOA reference architecture to define their solutions and map their specific products to the architectural models

Provides a reference for other standards bodies and Open Group work streams to use in the context of understanding SOA and providing a model for them to map against

The Open Group SOA Reference Architecture can be used in the following ways:

To understand the different elements of an SOA, including the key architectural elements in it and the key relationships between these elements

As a vehicle to provide traceability to and mapping between the common systems architecture (which the SOA reference architecture represents) and specific industry and organizational architectures

To provide a model and framework for determining and evaluating the set of relevant architectural aspects for designing an SOA

Further, it can be used as a guide to refining the SOA reference architecture (common systems architecture) into a domain (industry) or enterprise (organization) reference architecture and to instantiating it to produce a concrete architecture.

The Open Group SOA Reference Architecture can represent both abstract enterprise scale designs as well as concrete SOA implementations.

This SOA reference architecture uses a partially layered approach since one layer does not solely depend upon the adjacent layers. Layers are defined around sets of key architectural concerns and capabilities, the interaction protocols between layers, and the details within a layer using a set of architectural building blocks. There are five functional horizontal layers and four non-functional vertical layers that support various cross-cutting aspects of the SOA architectural style.

This SOA reference architecture consists of a set of conceptual elements, such as layers, architectural building blocks, and their mutual interactions. These elements need to be instantiated by making architectural and realization decisions on what vendor products, parts of products, standards, and protocols will be used to instantiate a given architectural building block. This allows and facilitates the creation of solutions based on

the reference architecture, at different levels; namely a logical down to the physical instantiation of a concrete architecture.

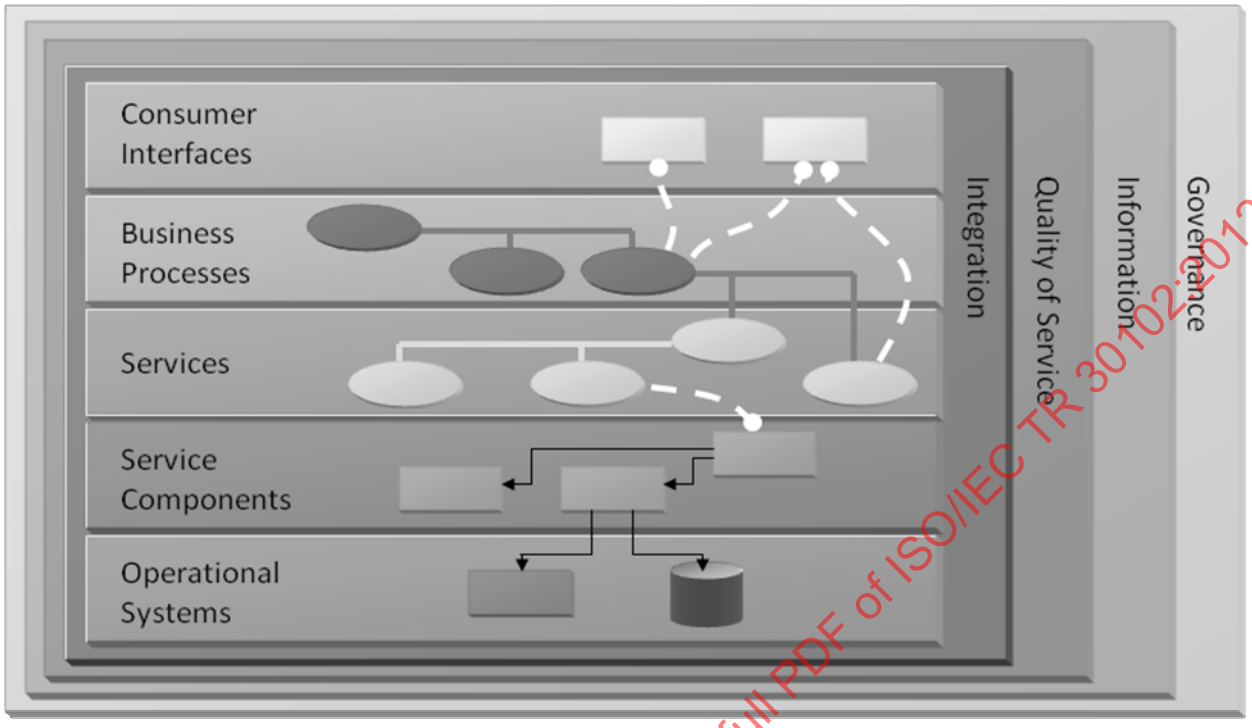

Figure A.1

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

The figure below shows an underlying middleware and infrastructure service categorization and how it applies to the SOA RA.

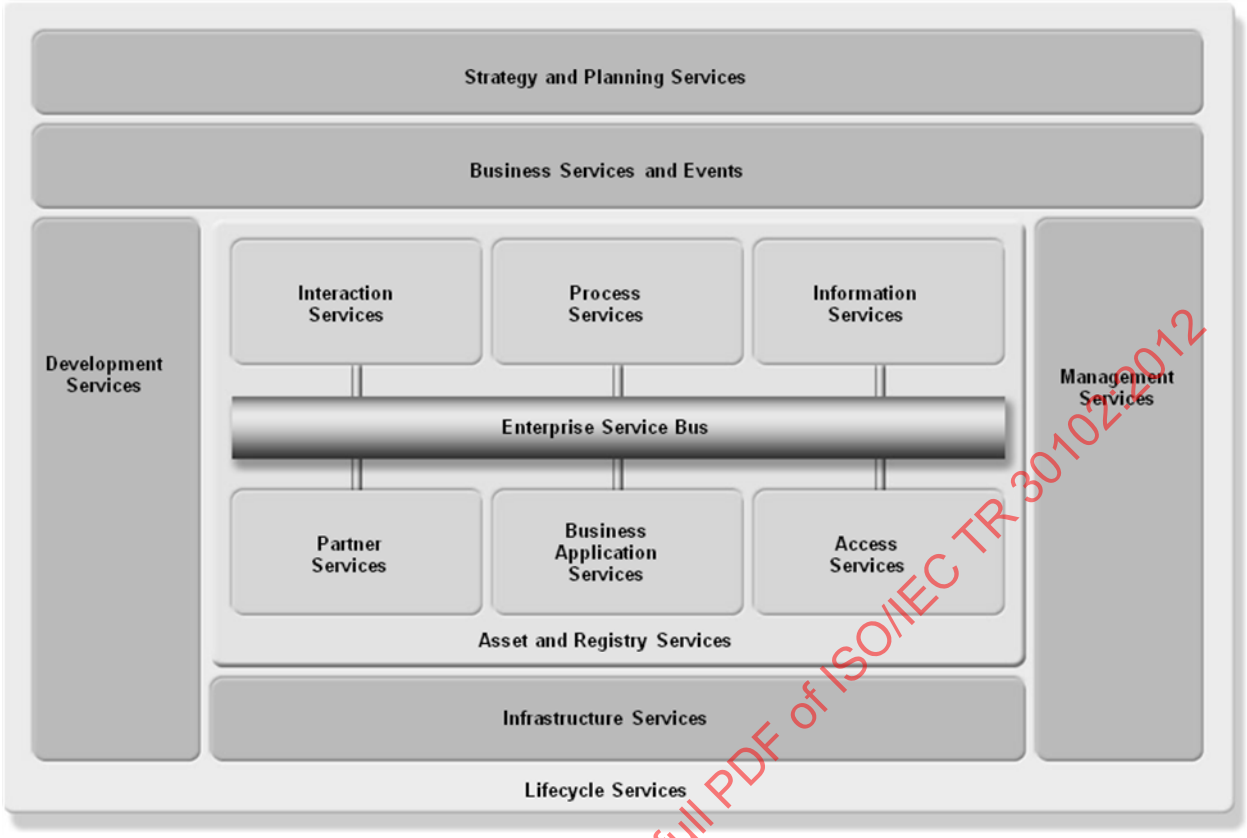

Figure A.2

The Open Group SOA Ontology (See [21]) is similar to the above OASIS Reference Model for SOA (see [19]) in that it captures a set of related concepts within the SOA space and explains what they are and how they relate to each other. The objectives are to facilitate understanding of these terms and concepts within the context of SOA, and potentially to facilitate model-driven implementation. The ontology is represented in OWL (Web Ontology Language, see [32[)] to enable automation and allow tools to process it; for example, reasoning applications could use the SOA ontology to assist in service consumer and provider matching, service value chain analysis, and impact analysis. The formal representation enables integration with other concerns such as business motivation modelling, business process modelling, information modelling, operations modelling, portfolio management, etc.

Note that The Open Group SOA Ontology and the OASIS Reference Model for SOA are very closely aligned, although some terms may represent different architectural views. The difference in expression or naming of concepts does not affect the basic understanding of SOA or the derivative architectures.

Annex B
(informative)

The OASIS SOA Reference Model and Reference Architecture

The OASIS Reference Model for SOA (see [19]) is intended to capture the "essence" of SOA, as well as provide a vocabulary and common understanding of SOA. The goals of the reference model include a common conceptual framework that can be used consistently across and between different SOA implementations, common semantics that can be used unambiguously in modeling specific SOA solutions, unifying concepts to explain and underpin a generic design template supporting a specific SOA, and definitions that should apply to all SOA. The reference model provides a normative reference that remains relevant for SOA as an abstract, powerful model, regardless of the inevitable technology changes that have influenced or will influence SOA deployment.

The OASIS Reference Architecture for SOA Foundation (see [26]) is a view-based abstract reference architecture foundation that models SOA from an ecosystem/paradigm perspective. It specifies three viewpoints; specifically, the *Service Ecosystem* viewpoint, the *Realizing SOAs* viewpoint, and the *Owning SOAs* viewpoint. Each of the associated views that are obtained from these three viewpoints is briefly described below. Since it is an abstract and foundational reference architecture, it does not contain the level of specificity required to directly implement SOA-based systems. It does provide UML models and architectural implications for each of the views useful in guiding other architecture work, including other reference architectures, as architects become more enterprise and/or solution-oriented.

The *Service Ecosystem* view contains models that are intended to capture how SOA integrates with and supports the service model from the perspective of the people who perform their tasks and achieve their goals as mediated by SOAs. Since the Service Ecosystem viewpoint (on which this view is based) emphasizes the use of SOA to allow people to access and provide services that cross ownership boundaries, it is explicit about those boundaries and what it means to cross an ownership boundary.

The *Realizing SOAs* view contains models for description of, visibility of, interaction with, and policies for services.

The *Owning SOAs* view contains models for securing, managing, governing, and testing SOA-based systems.

Annex C
(informative)

OMG SOA / Modeling Language

Business and IT architects also employ methodologies for modeling and building architectures. As such, architectural methodologies have emerged with the advent of Model Driven Architecture (MDA, see [31]), a technical product of the OMG. For working with SOA and using the Unified Modeling Language (UML, see [30]) as the primary syntax, the OMG SoaML specification (see [31]) provides guidance and a metamodel to help architects and other strategic thinkers link the design of real world SOA-based systems into their architecture work.

SoaML is an OMG standard that defines extensions to UML for services modeling and provides functional, component, and service-oriented modeling capabilities. Each of these modeling approaches provides different, enhanced capabilities for dealing with cohesion and coupling in complex systems. SoaML extends UML in order to provide additional capabilities for managing cohesion and coupling afforded by an SOA style. SoaML is applicable across a broad range of domains and levels of abstraction from business services to detailed IT services. Using a common language for these different purposes simplifies systems modeling and integration of separate concerns in order to enable business agility, which can be represented with business architecture models such as BMM and BPMN. SoaML can be viewed as supporting instantiation of the OASIS Reference Model for SOA (see [19]) that provides a concrete platform for services modeling integrated with UML and supporting OMG MDA.

The purpose of the SoaML standard is to address service modeling, not methodologies for determining what the services model should be, or how it would be used in any particular context. The standard is intended to be sufficiently detailed to define platform-independent SOA models (PIM) that can be transformed into platform-specific models (PSM) for particular technical architectures as described by the OMG MDA. The scope of SoaML does not cover SOA governance or compliance, quality of services (policy, trust, performance, etc.), message delivery reliability, wire-level protocols, service brokering, publishing discovery, etc. Rather, it is expected that SoaML will be integrated with other standards that already address these aspects, or be extended over time to support them directly. The intent of SoaML was to provide a foundation for integration, interoperability, and extension.

The fundamental element of SoaML is the participant, representing a service consumer and/or provider. Participants express their goals, needs, and expectations through requests for services as defined by service interfaces or service contracts. Other participants express their value propositions, capabilities, and commitments through services. Participants are then assembled into service value chains where participant requests are connected to the compatible services of other participants through service channels through which they interact. SoaML uses facilities of UML to define the services interfaces and method behaviors for carrying out and using services. SoaML also defines autonomous agents that can choreograph participants in a service value chain while adapting to the changing needs of the community of collaborating participants. SoaML provides a means of defining milestones that indicate the achievement of progress toward achieving the desired real-world effect of the services value chain, and for evaluating different approaches to achieving progress by different participants.

Annex D
(informative)

China's Technical Reference Architecture for SOA Solutions

NOTE Annex D contains national body standard, 'Information technology-General technical requirement of Service Oriented Architecture (SOA) based application', considered during the development of this report and are included for context and reference. No comments on this Annex were requested, processed, or addressed in the development of this Annex and TR, therefore was no consensus developed on this annex.

D.1 Technical Reference Architecture for SOA Solutions

The Technical Reference Architecture defines the general technical principles for SOA solutions. The following diagram illustrates the potential scope to which the principles of SOA would apply:

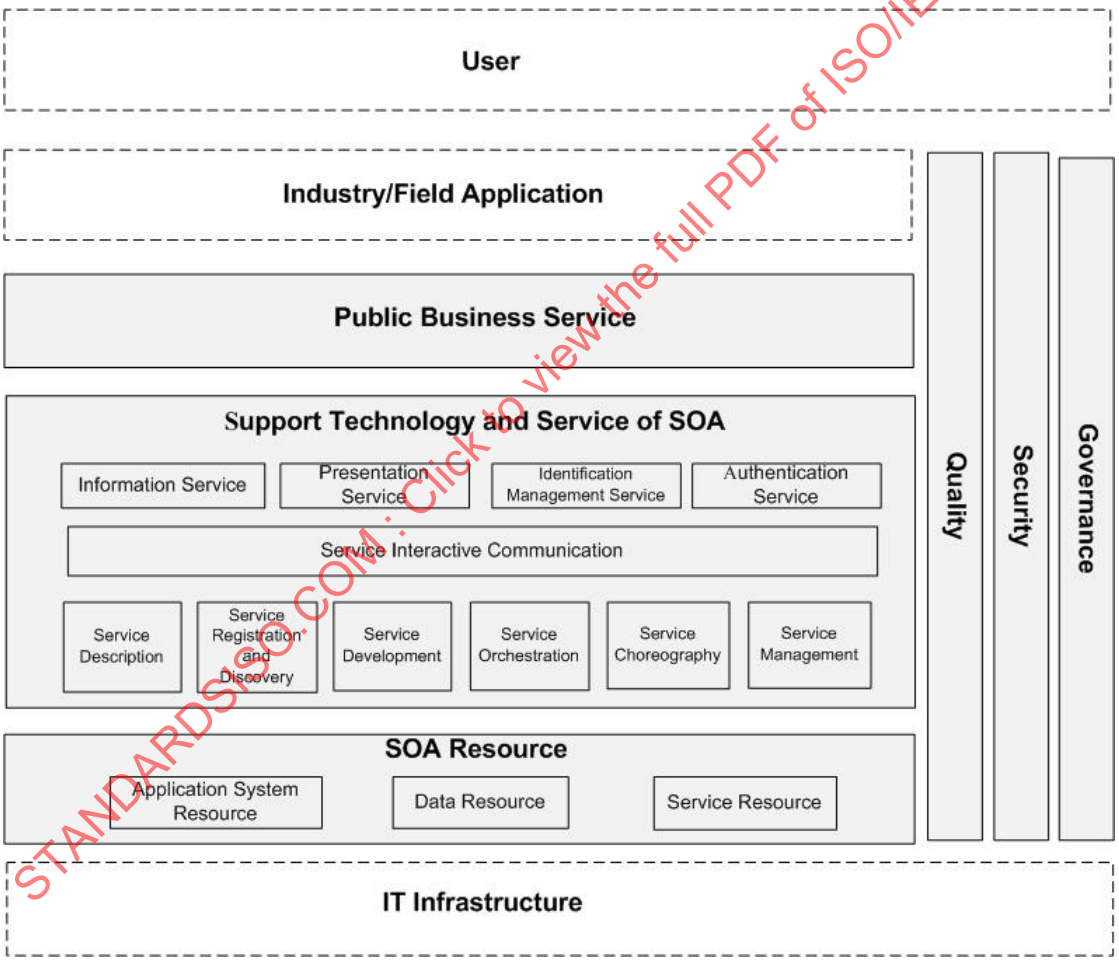

Figure D.1 — Technical Reference Architecture for SOA Solutions

Technical Reference Architecture for SOA Solutions covers construction, operation and management of SOA solutions.

Technical Reference Architecture for SOA Solutions includes 9 major parts:

1. IT Infrastructure is the operational environment of the SOA solution. This operational envrionement is configurable and scalable.
2. SOA Resource is the IT resources used to implement SOA solutions, such as application systems, data, and existing services. These resources exist in companies, government departments and other organizations. They are the original source for constructing SOA solutions.
3. Support Technology and Service of SOA is the general terms of fundamental technical abilities and technical services which support SOA solutions.
4. Public Business Service is a set of services which are industry-oriented/field-oriented, reusable, and functional.for certain business
5. Industry/Field Application is IT systems which are user-oriented, specific industry-based or specific field-based.
6. User is the general terms of humans, systems, devices and other services which use SOA solutions.
7. Quality is the level in which SOA solutions satisfy users' requirement or expectation.
8. Security is the the general terms or mechanisms and strategies which ensure the safe operation of SOA solutions.
9. Governance is the management and control strategy,and the mechanism which is aim at SOA solution. It covers the entire life circle of SOA solutions.

D.1.1 Requirment for SOA Resource in SOA Solutions

This chapter describes typical requirement for resource in SOA solutions.

D.1.1.1 Requirement for Application System Resource

Application system resource should satisfy the following requirements:
operating in homogeneous or heterogeneous environments
providing specific function to deal business
providing standard interface or adapter for accessibility.

D.1.1.2 Requirement for Data Resource

Data resources should meet the following requirements:
corresponding to one of the following descriptions:

1. the information of storage data in the business system
2. the message formats definition and message content
3. other formats files or services definitions

providing standard interface for accessibility.

D.1.1.3 Requirement for Service Resource

Service resources should meet the following requirements:

a) It is one functional bussiness element has follow characteristics:
   1.standardized description
   2.prividing external address for accessibility
   3.reusage
b) existing before the construction of SOA solution, being one part of the IT resource.
c) being provided by the organization itself or by third-party organization.

D.2 Requirment for Support Technology and Services in SOA Solutions

This chapter describes the basic technical capabilities and basic technical services for supporting SOA solutions. The typical elements includes service description and discovery, service management, service composition, service flow scheduling, service development, service security, service interactive communications, information services, display services, identity management services and authorization services.

D.2.1 Requirement for Service Description Capability

The service description capability should meet the following requirements:

a) providing a standard information model, access interface, and description of related properties of services and resources.
b) describing the related specific technical standards for the service description.

D.2.2 Requirement for Service Registration and Discovery Capability

Service registration and Discovery capability should meet the following requirements:

a) providing an interface for service registration and access so that services and resources can be registered, searched and found.
b) providing the related specific technical standards for service registration and service discovery.

D.2.3 Requirement for Service Development Capability

Service development capability should meet the following requirements:

providing development, configuration, debugging and testing environments for the construction of new services.

service encapsulation of existing application systems or data resources.

providing the related specific technical standard for service development.

D.2.4 Requirement for Service Orchestration Capability

Service orchestration capability should meet the following requirements:

a) calling a series of services according to a certain logical sequence to form a coarser granularity service.
b) providing container environments for service orchestration.
c) providing the related specific technical standards for service orchestration.

D.2.5 Requirement for Service Choreography Capability

Service choreography capability should meet the following requirements:

building new services by composing other services via service process modeling and / or service choreography. The new services should satisfy business processes.

providing process execution engines. The engine should include support for interpretation, execution, control, management and other functions for deployed business process scripts.

providing the related specific technical standards for service choreography.

D.2.6 Requirement for Service Management Capability

Service management capability should meet the following requirements:

a) real-time monitoring, early warning, alerting, and other related management operations for the service status.
b) Providing the related specific technical standard of service management.

D.2.7 Requirement for Service Interactive Communication Capability

Service interactive communication capability should meet the following requirements:

a) providing access, communication, routing and switching of services.
b) integraing with service management.
c) providing service interaction mechanisms and quality assurance.
d) providing the related specific technical standards for service interactive communication.

D.2.8 Requirement for Information Service

Information Service should meet the following requirements:

a) providing for the collecting, cataloging, publishing, searching, etc of information.
b) providing related specific technical standards for information Services.

D.2.9 Requirement for Presentation Service

Presentation service should meet the following requirements:

a) providing a set of complete, multi-channel accessed human-computer interaction functions.
b) providing the related specific technical standards for presentation services.

D.2.10 Requirement for Identification Management Service

Identification management service should meet the following requirements:

a) providing a set of extensible management functions for organization, personnel, roles, authentication, etc.
b) providing the related specific technical standards for identification management services.

D.2.11 Requirement for Authentication Service

Authentication service should meet the following requirements:

a) providing access control based on identification management services.
b) providing the related specific technical standards for authentication services.

D.3 Requirement for Public Business Service in SOA Solutions

This chapter defines public business service capabilities. During the implementation process for SOA solution systems, public business services which have industry/field specific features should be gradualy introduced. Public business services should support the features of the SOA solution.

Public business services should meet the following requirements:

a) satisfying every essential function of the service and realize part of the industry/field business functions.
b) being strongly reusable within a certain area.
c) supporting the related standards or criteria of industry and field standards.

D.4 Requirement for Quality in SOA Solutions

This chapter defines the quality capabilities for SOA solutions apart from functional capabilities. It covers general quality capabilities and service quality capabilities of a solution system.

D.4.1 General Requirement

Apart from the functionality requirements defined in chaper 5.1, 5.2 and 5.3, SOA solution should also meet the following requirement for quality:

a) Reliability
b) Ease of Use
c) Efficiency
d) Maintainability
e) Portability

Note: The specific indicators and parameters are given in other standards.

D.4.2 Requirement for Service Quality

Service quality should at least meet the following requirements:
Reasonable Service Granularity
Loose Coupling
Reusability
Scalability
Interoperability

Note: The specific indicators and parameters are given in other standards.

D.5 Requirement for Security in SOA Solutions

This chapter describes the requirement for Security in SOA solutions:

a) providing access control and authentication management of related resources.
b) providing secure interaction services, including the security protection measures during the transmission process to prevent information tampering, leaking, etc.
c) providing security for the information/data of the services. It should provide encryption and decryption, signature, data integrity verification and other services for information / data and other resources.
d) providing auditing and logging services, including access to audit logs, history and tracking logs, and statistical services.
e) up to the sprcific technical standard related service security.

D.6 Requirement for Governance in SOA Solutions

This chapter describes the requirment for governance in the entire life cycle of SOA solution, includes general requirment and requirement for service governance.

D.6.1 General Requirement

SOA solutions should support general IT governance capabilities for the entire lifecycle of IT systems.

D.6.2 Requirement for Service Governance

Service governance should meet the following requirements:

a) Service Planning Governance
b) Service Development Management
c) Service Operation Management
d) Service Optimization