Reference Model for Service Oriented Architecture 1.0

OASIS Standard, 12 October 2006

Document identifier:

soa-rm

Location:

http://docs.oasis-open.org/soa-rm/v1.0/

Editors:

C. Matthew MacKenzie, Adobe Systems Incorporated, mattm@adobe.com
Ken Laskey, MITRE Corporation, klaskey@mitre.org
Francis McCabe, Fujitsu Laboratories of America Limited, frankmccabe@mac.com
Peter F Brown, peter@justbrown.net
Rebekah Metz, Booz Allen Hamilton, metz_rebekah@bah.com

Abstract:

This Reference Model for Service Oriented Architecture is an abstract framework for understanding significant entities and relationships between them within a service-oriented environment, and for the development of consistent standards or specifications supporting that environment. It is based on unifying concepts of SOA and may be used by architects developing specific service oriented architectures or in training and explaining SOA.

A reference model is not directly tied to any standards, technologies or other concrete implementation details. It does seek to provide a common semantics that can be used unambiguously across and between different implementations. The relationship between the Reference Model and particular architectures, technologies and other aspects of SOA is illustrated in Figure 1.

While service-orientation may be a popular concept found in a broad variety of applications, this reference model focuses on the field of software architecture. The concepts and relationships described may apply to other "service" environments; however, this specification makes no attempt to completely account for use outside of the software domain.

Status:

This document is updated periodically on no particular schedule. Send comments to the editor(s).
Committee members should send comments on this specification to the soa-rm@lists.oasis-open.org list. Others should visit the SOA-RM TC home page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, and record comments using the web form available there.
For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the SOA-RM TC web page at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm
The errata page for this specification is at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Table of Contents

1 Introduction 4
1.1 What is a reference model 4
1.2 A Reference Model for Service Oriented Architectures 4
1.3 Audience 5
1.4 Guide to using the reference model 6
1.5 Notational Conventions 6
1.5.1 How to interpret concept maps. 6
1.6 Relationships to Other Standards 7
2 Service Oriented Architecture 8
2.1 What is Service Oriented Architecture? 8
2.1.1 A worked Service Oriented Architecture example 9
2.2 How is Service Oriented Architecture different? 10
2.3 The Benefits of Service Oriented Architecture 11
3 The Reference Model 12
3.1 Service 12
3.2 Dynamics of Services 13
3.2.1 Visibility 13
3.2.2 Interacting with services 15
3.2.3 Real World Effect 18
3.3 About services 19
3.3.1 Service description 20
3.3.2 Policies and Contracts 22
3.3.3 Execution context 24
4 Conformance Guidelines 26
5 References 27
5.1 Normative 27
5.2 Non-Normative 27
A. Glossary 28
B. Acknowledgments 31

1 Introduction

The notion of Service Oriented Architecture (SOA) has received significant attention within the software design and development community. The result of this attention is the proliferation of many conflicting definitions of SOA. Whereas SOA architectural patterns (or *reference architectures*) may be developed to explain and underpin a generic design template supporting a specific SOA, a **reference model** is intended to provide an even higher level of commonality, with definitions that should apply to *all* SOA.

1.1 What is a reference model

A reference model is an abstract **framework** for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details.

As an illustration of the relationship between a reference model and the architectures that can derive from such a model, consider what might be involved in modeling what is important about residential housing. In the context of a reference model, we know that concepts such as eating areas, hygiene areas and sleeping areas are all important in understanding what goes into a house. There are relationships between these concepts, and constraints on how they are implemented. For example, there may be physical separation between eating areas and hygiene areas.

The role of a **reference architecture** for housing would be to identify abstract solutions to the problems of providing housing. A general pattern for housing, one that addresses the needs of its occupants in the sense of, say, noting that there are bedrooms, kitchens, hallways, and so on is a good basis for an abstract reference architecture. The concept of eating area is a reference model concept, a kitchen is a realization of eating area in the context of the reference architecture.

There may be more than one reference architecture that addresses how to design housing; for example, there may be a reference architecture to address the requirements for developing housing solutions in large apartment complexes, another to address suburban single family houses, and another for space stations. In the context of high density housing, there may not be a separate kitchen but rather a shared cooking space or even a communal kitchen used by many families.

An actual – or concrete – architecture would introduce additional elements. It would incorporate particular architectural styles, particular arrangements of windows, construction materials to be used and so on. A blueprint of a particular house represents a specific architecture as it applies to a proposed or actually constructed dwelling.

The reference model for housing is, therefore, at least three levels of abstraction away from a physical entity that can be lived in. The purpose of a reference model is to provide a common conceptual framework that can be used consistently across and between different implementations and is of particular use in modeling specific solutions.

1.2 A Reference Model for Service Oriented Architectures

The goal of this reference model is to define the essence of service oriented architecture, and emerge with a vocabulary and a common understanding of SOA. It provides a normative reference that remains relevant for SOA as an abstract and powerful model, irrespective of the various and inevitable technology evolutions that will influence SOA deployment.

Figure 1 shows how a reference model for SOA relates to other distributed systems architectural inputs. The concepts and relationships defined by the reference model are intended to be the basis for describing references architectures and patterns that will define more specific categories of SOA designs. Concrete architectures arise from a combination of reference architectures, architectural patterns and additional requirements, including those imposed by technology environments.

Architecture must account for the goals, motivation, and requirements that define the actual problems being addressed. While reference architectures can form the basis of classes of solutions, concrete architectures will define specific solution approaches.

Architecture is often developed in the context of a pre-defined environment, such as the protocols, profiles, specifications, and standards that are pertinent.

SOA implementations combine all of these elements, from the more generic architectural principles and infrastructure to the specifics that define the current needs, and represent specific implementations that will be built and used in an operational environment.

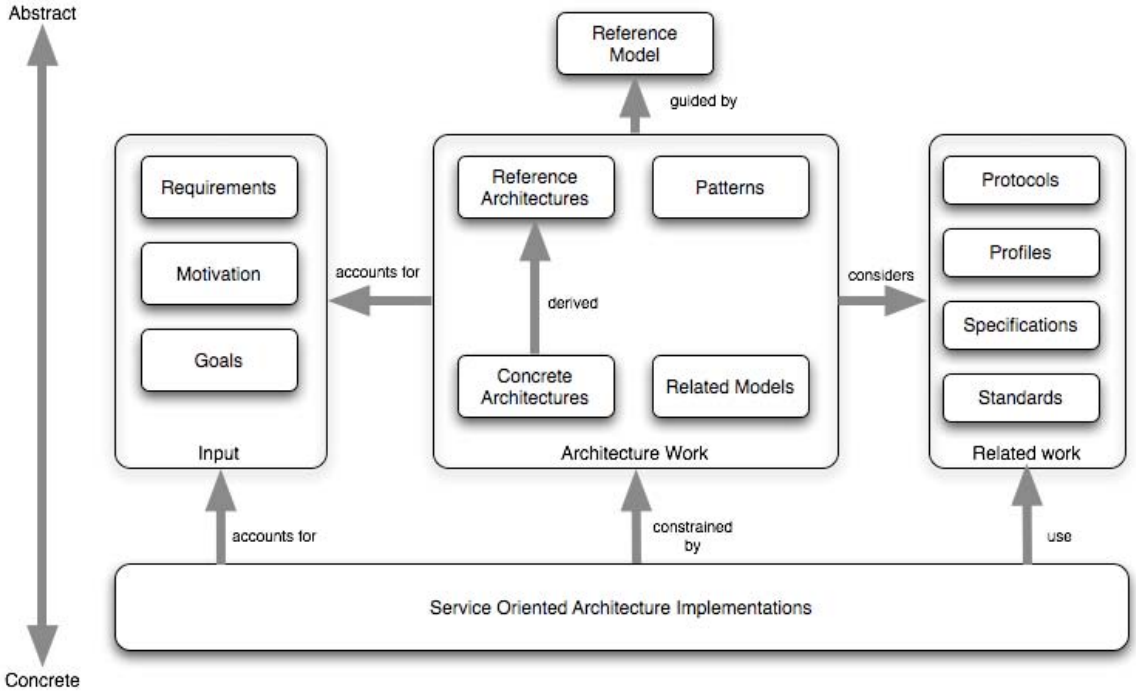


Figure 1 How the Reference Model relates to other work

1.3 Audience

The intended audiences of this document include non-exhaustively:

- Architects and developers designing, identifying or developing a system based on the service-oriented paradigm.
- Standards architects and analysts developing specifications that rely on service oriented architecture concepts.
- Decision makers seeking a "consistent and common" understanding of service oriented architectures.
- Users who need a better understanding of the concepts and benefits of service oriented architecture.

1.4 Guide to using the reference model

New readers are encouraged to read this reference model in its entirety. Concepts are presented in an order that the authors hope promote rapid understanding.

This section introduces the conventions, defines the audience and sets the stage for the rest of the document. Non-technical readers are encouraged to read this information as it provides background material necessary to understand the nature and usage of reference models.

Section 2 introduces the concept of SOA and identifies some of the ways that it differs from previous paradigms for distributed systems. Section 2 offers guidance on the basic principles of service oriented architecture. This can be used by non-technical readers to gain an explicit understanding of the core principles of SOA and by architects as guidance for developing specific service oriented architectures.

Section 3 introduces the Reference Model for SOA. In any framework as rich as SOA, it is difficult to avoid a significant amount of cross referencing between concepts. This makes presentation of the material subject to a certain amount of arbitrariness. We resolve this by introducing the concept of service itself, then we introduce concepts that relate to the dynamic aspects of service and finally we introduce those concepts that refer to the meta-level aspects of services such as service description and policies as they apply to services.

Section 4 addresses compliance with this reference model.

The glossary provides definitions of terms that are relied upon within the reference model specification but do not necessarily form part of the specification itself. Terms that are defined in the glossary are marked in **bold** at their first occurrence in the document.

Note that while the concepts and relationships described in this reference model may apply to other "service" environments, the definitions and descriptions contained herein focus on the field of **software architecture** and make no attempt to completely account for use outside of the software domain. Examples included in this document that are taken from other domains are used strictly for illustrative purposes.

1.5 Notational Conventions

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC2119].

References are surrounded with **[square brackets and are in bold text]**.

1.5.1 How to interpret concept maps.

Concepts maps are used within this document. There is no normative convention for interpreting Concept maps and other than described herein, no detailed information can be derived from the concept maps herein.

Figure 2 A basic concept map

As used in this document a line between two concepts represents a relationship, where the relationship is not labeled but rather is described in the text immediately preceding or following the figure. The arrow on a line indicates an asymmetrical relationship, where the concept to which the arrow points (Concept 2 in Figure 2) can be interpreted as depending in some way on

the concept from which the line originates (Concept 1). The text accompanying each graphic describes the nature of each relationship.

1.6 Relationships to Other Standards

Due to its nature, this reference model may have an implied relationship with any group that:

- Considers its work "service oriented";
- Makes (publicly) an adoption statement to use the Reference Model for SOA as a base or inspiration for their work; and
- Standards or technologies that claim to be service oriented.

The reference model does not endorse any particular service-oriented architecture, or attest to the validity of third party reference model conformance claims.

2 Service Oriented Architecture

2.1 What is Service Oriented Architecture?

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed **capabilities** that may be under the control of different ownership domains.

In general, entities (people and organizations) create capabilities to solve or support a solution for the problems they face in the course of their business. It is natural to think of one person's needs being met by capabilities offered by someone else; or, in the world of distributed computing, one computer agent's requirements being met by a computer agent belonging to a different owner.

There is not necessarily a one-to-one correlation between needs and capabilities; the granularity of needs and capabilities vary from fundamental to complex, and any given need may require the combining of numerous capabilities while any single capability may address more than one need. The perceived value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs.

Visibility, interaction, and effect are key concepts for describing the SOA paradigm. **Visibility** refers to the capacity for those with needs and those with capabilities to be able to see each other. This is typically done by providing descriptions for such aspects as functions and technical requirements, related constraints and policies, and mechanisms for access or response. The descriptions need to be in a form (or can be transformed to a form) in which their syntax and **semantics** are widely accessible and understandable.

Whereas visibility introduces the possibilities for matching needs to capabilities (and vice versa), **interaction** is the activity of using a capability. Typically mediated by the exchange of messages, an interaction proceeds through a series of information exchanges and invoked actions. There are many facets of interaction; but they are all grounded in a particular **execution context** – the set of technical and business elements that form a path between those with needs and those with capabilities. This permits service providers and consumers to interact and provides a decision point for any policies and contracts that may be in force.

The purpose of using a capability is to realize one or more **real world effects**. At its core, an interaction is "an act" as opposed to "an object" and the result of an interaction is an effect (or a set/series of effects). This effect may be the return of information or the change in the state of entities (known or unknown) that are involved in the interaction.

We are careful to distinguish between *public* actions and *private* actions; private actions are inherently unknowable by other parties. On the other hand, public actions result in changes to the *state* that is shared between at least those involved in the current execution context and possibly shared by others. Real world effects are, then, couched in terms of changes to this **shared state**.

The expected real world effects form an important part of the decision on whether a particular capability matches similarly described needs. At the interaction stage, the description of real world effects establishes the expectations of those using the capability. Note, it is not possible to describe every effect from using a capability. A cornerstone of SOA is that capabilities can be used without needing to know all the details.

This description of SOA has yet to mention what is usually considered the central concept: the **service**. The noun "service" is defined in dictionaries as "The performance of work (a function) by one for another." However, service, as the term is generally understood, also combines the following related ideas:

- The capability to perform work for another
- The specification of the work offered for another
- The offer to perform work for another

These concepts emphasize a distinction between a capability and the ability to bring that capability to bear. While both needs and capabilities exist independently of SOA, **in SOA, services are the mechanism by which needs and capabilities are brought together.**

SOA is a means of organizing solutions that promotes reuse, growth and interoperability. It is not itself a solution to domain problems but rather an organizing and delivery paradigm that enables one to get more value from use both of capabilities which are locally "owned" and those under the control of others. It also enables one to express solutions in a way that makes it easier to modify or evolve the identified solution or to try alternate solutions. SOA does not provide any domain elements of a solution that do not exist without SOA.

Note that while an SOA service brings together needs and capabilities, the provider of the underlying capability may not be the same entity that eventually provides the service which accesses that capability. In reality, the entity with the domain expertise to create, maintain, and evolve a given capability may not have the expertise or the desire to create, maintain, and evolve its service access.

The concepts of visibility, interaction, and effect apply directly to services in the same manner as these were described for the general SOA paradigm. Visibility is promoted through the **service description** which contains the information necessary to interact with the service and describes this in such terms as the service inputs, outputs, and associated semantics. The service description also conveys what is accomplished when the service is invoked and the conditions for using the service.

In general, entities (people and organizations) offer capabilities and act as **service providers**. Those with needs who make use of services are referred to as **service consumers**. The service description allows prospective consumers to decide if the service is suitable for their current needs and establishes whether a consumer satisfies any requirements of the service provider.

(Note, service providers and service consumers are sometimes referred to jointly as **service participants**.)

In most discussions of SOA, the terms "loose coupling" and "coarse-grained" are commonly applied as SOA concepts, but these terms have intentionally not been used in the current discussion because they are subjective trade-offs and without useful metrics. In terms of needs and capabilities, granularity and coarseness are usually relative to detail for the level of the problem being addressed, e.g. one that is more strategic vs. one down to the algorithm level, and defining the optimum level is not amenable to counting the number of interfaces or the number or types of information exchanges connected to an interface.

Note that although SOA is commonly implemented using Web services, services can be made visible, support interaction, and generate effects through other implementation strategies. Web service-based architectures and technologies are specific and concrete. While the concepts in the Reference Model apply to such systems, Web services are too solution specific to be part of a general reference model.

2.1.1 A worked Service Oriented Architecture example

> An electric utility has the capacity to generate and distribute electricity (the underlying capability). The wiring from the electric company's distribution grid (the service) provides the means to supply electricity to support typical usage for a residential consumer's house (service functionality), and a consumer accesses electricity generated (the output of invoking the service) via a wall outlet (service interface). In order to use the electricity, a consumer needs to understand what type of plug to use, what is the voltage of the supply, and possible limits to the load; the utility presumes that the customer will only connect devices that are compatible with the voltage provided and load supported; and the consumer in turn assumes that compatible consumer devices can be connected without damage or harm (service technical assumptions).

A residential or business user will need to open an account with the utility in order to use the supply (service constraint) and the utility will meter usage and expects the consumer to pay for use at the rate prescribed (service policy). When the consumer and utility agree on constraints and polices (service contract), the consumer can receive electricity using the service as long as the electricity distribution grid and house connection remain intact (e.g. a storm knocking down power lines would disrupt distribution) and the consumer can have payment sent (e.g. a check by mail or electronic funds transfer) to the utility (reachability).

Another person (for example, a visitor to someone else's house) may use a contracted supply without any relationship with the utility or any requirement to also satisfy the initial service constraint (i.e. reachability only requires intact electricity distribution) but would nonetheless be expected to be compatible with the service interface.

In certain situations (for example, excessive demand), a utility may limit supply or institute rolling blackouts (service policy). A consumer might lodge a formal complaint if this occurred frequently (consumer's implied policy).

If the utility required every device to be hardwired to its equipment, the underlying capability would still be there but this would be a very different service and have a very different service interface.

2.2 How is Service Oriented Architecture different?

Unlike Object Oriented Programming paradigms, where the focus is on packaging data with operations, the central focus of Service Oriented Architecture is the task or business function – getting something done.

This distinction manifests itself in several ways:

- OO has intentional melding of methods to a given data object. The methods can be thought of as a property of the object. For SOA, one can think of the services as being the access to methods but the actual existence of methods and any connection to objects is incidental.
- To use an object, it must first be instantiated while one interacts with a service where it exists.
- An object exposes structure but there is no way to express semantics other than what can be captured as comments in the class definition. SOA emphasizes the need for clear semantics.

Both OO and SOA are as much a way of thinking about representing things and actions in the world as these are about the specifics of building a system. The important thing is understanding and applying the paradigm. So the question is not “what is a service?” any more than it is “what is an object?” Anything can be a service in the same way anything can be an object. The challenge is to apply the paradigms to enhance clarity and get things done. SOA provides a more viable basis for large scale systems because it is a better fit to the way human activity itself is managed – by delegation.

How does this paradigm of SOA differ from other approaches to organizing and understanding Information Technology assets? Essentially, there are two areas in which it differs both of which shape the framework of concepts that underlie distributed systems.

First, SOA reflects the reality that ownership boundaries are a motivating consideration in the architecture and design of systems. This recognition is evident in the core concepts of visibility, interaction and effect.

However, SOA does not itself address all the concepts associated with ownership, ownership domains and actions communicated between legal peers. To fully account for concepts such as trust, business transactions, authority, delegation and so on – additional conceptual frameworks and architectural elements are required. Within the context of SOA, these are likely to be represented and referenced within **service descriptions** and **service interfaces**. The presence

of service descriptions and service interfaces provides a ready location for including such references and thus facilitates reuse of externally developed frameworks and interoperability among systems availing themselves of this reuse.

Second, SOA applies the lessons learned from commerce to the organization of IT assets to facilitate the matching of capabilities and needs. That two or more entities come together within the context of a single interaction implies the exchange of some type of value. This is the same fundamental basis as trade itself, and suggests that as SOAs evolve away from interactions defined in a point-to-point manner to a marketplace of services; the technology and concepts can scale as successfully as the commercial marketplace.

2.3 The Benefits of Service Oriented Architecture

The main drivers for SOA-based architectures are to facilitate the manageable growth of large-scale enterprise systems, to facilitate Internet-scale provisioning and use of services and to reduce costs in organization to organization cooperation.

The value of SOA is that it provides a simple scalable paradigm for organizing large networks of systems that require interoperability to realize the value inherent in the individual components. Indeed, SOA is scalable because it makes the fewest possible assumptions about the network and also minimizes any trust assumptions that are often implicitly made in smaller scale systems.

An architect using SOA principles is better equipped, therefore, to develop systems that are scalable, evolvable and manageable. It should be easier to decide how to integrate functionality across ownership boundaries. For example, a large company that acquires a smaller company must determine how to integrate the acquired IT infrastructure into its overall IT portfolio.

Through this inherent ability to scale and evolve, SOA enables an IT portfolio which is also adaptable to the varied needs of a specific problem domain or process architecture. The infrastructure SOA encourages is also more agile and responsive than one built on an exponential number of pair-wise interfaces. Therefore, SOA can also provide a solid foundation for business agility and adaptability.

3 The Reference Model

Figure 3 illustrates the principal concepts this reference model defines. The relationships between them are developed as each concept is defined in turn.

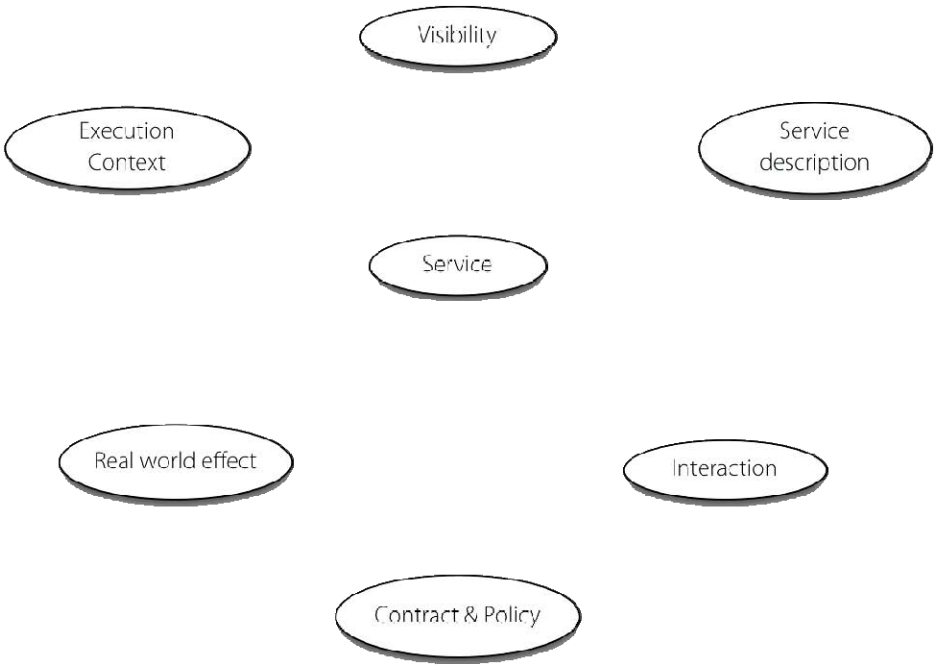


Figure 3 Principal concepts in the Reference Model

3.1 Service

A **service** is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by an entity – the **service provider** – for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider.

A service is accessed by means of a service interface (see Section 3.3.1.4), where the interface comprises the specifics of how to access the underlying capabilities. There are no constraints on what constitutes the underlying capability or how access is implemented by the service provider. Thus, the service could carry out its described functionality through one or more automated and/or manual processes that themselves could invoke other available services.

A service is opaque in that its implementation is typically hidden from the **service consumer** except for (1) the information and behavior models exposed through the service interface and (2) the information required by service consumers to determine whether a given service is appropriate for their needs.

The consequence of invoking a service is a realization of one or more real world effects (see Section 3.2.3). These effects may include:

1. information returned in response to a request for that information,
2. a change to the shared state of defined entities, or

3. some combination of (1) and (2).

Note, the service consumer in (1) does not typically know how the information is generated, e.g. whether it is extracted from a database or generated dynamically; in (2), it does not typically know how the state change is effected.

The service concept above emphasizes a distinction between a capability that represents some functionality created to address a need and the point of access where that capability is brought to bear in the context of SOA. It is assumed that capabilities exist outside of SOA. In actual use, maintaining this distinction may not be critical (i.e. the service may be talked about in terms of being the capability) but the separation is pertinent in terms of a clear expression of the nature of SOA and the value it provides.

3.2 Dynamics of Services

From a dynamic perspective, there are three fundamental concepts that are important in understanding what is involved in interacting with services: the visibility between service providers and consumers, the interaction between them, and the real world effect of interacting with a service.

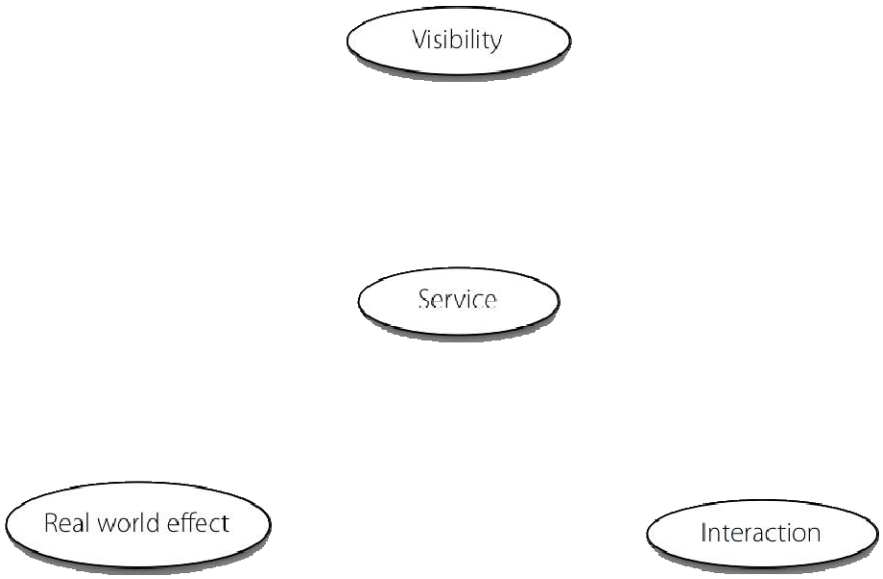


Figure 4 Concepts around the dynamics of service

3.2.1 Visibility

For a service provider and consumer to interact with each other they have to be able to 'see' each other. This is true for any consumer/provider relationship – including in an application program where one program calls another: without the proper libraries being present the function call cannot complete. In the case of SOA, visibility needs to be emphasized because it is not necessarily obvious how service participants *can* see each other.

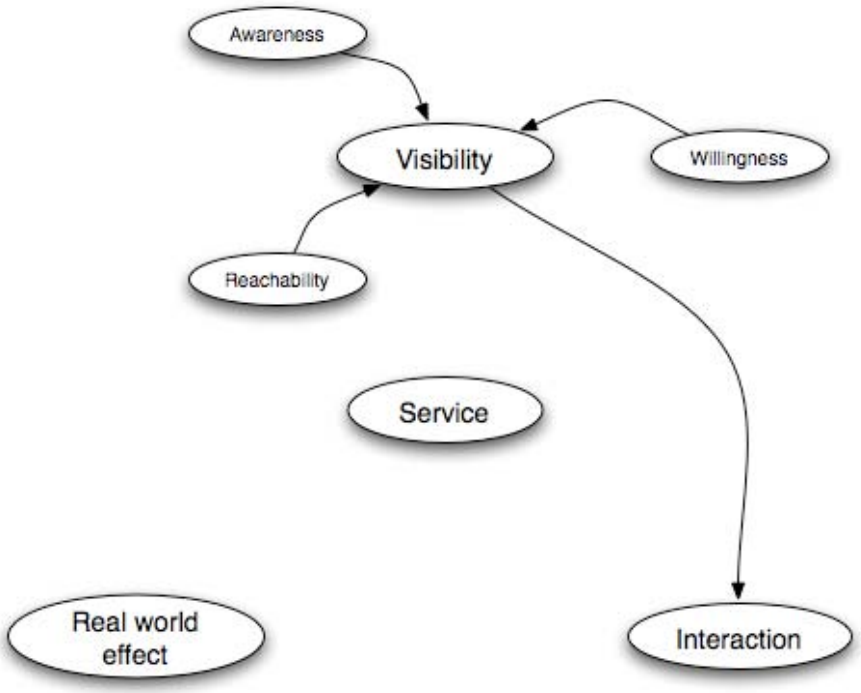


Figure 5 Concepts around Visibility

Visibility is the relationship between service consumers and providers that is satisfied when they are able to interact with each other. Preconditions to visibility are **awareness**, **willingness** and **reachability**. The initiator in a service interaction MUST be aware of the other parties, the participants MUST be predisposed to interaction, and the participants MUST be able to interact.

3.2.1.1 Awareness

Both the service provider and the service consumer MUST have information that would lead them to know of the other's existence. Technically, the prime requirement is that the *initiator* of a service interaction has knowledge of the responder. The fact of a successful initiation is often sufficient to inform the responder of the other's existence.

Awareness is a state whereby one party has knowledge of the existence of the other party. Awareness does not imply willingness or reachability. Awareness of service offerings is often effected by various *discovery* mechanisms. For a service consumer to discover a service, the service provider must be capable of making details of the service (notably service description and policies) available to potential consumers; and consumers must be capable of becoming aware of that information. Conversely, the service provider may want to discover likely consumers and would need to become aware of the consumer's description. In the following, we will discuss awareness in terms of service visibility but the concepts are equally valid for consumer visibility.

Service awareness requires that the **service description** and **policy** – or at least a suitable subset thereof – be available in such a manner and form that, directly or indirectly, a potential consumer is aware of the existence and capabilities of the service. The extent to which the description is "pushed" by the service provider, "pulled" by a potential consumer, subject to a probe or another method, will depend on many factors.

For example, a service provider may advertise and promote their service by either including it in a service directory or broadcasting it to all consumers; potential consumers may broadcast their particular service needs in the hope that a suitable service responds with a proposal or **offer,** or a service consumer might also probe an entire network to determine if suitable services exist. When the demand for a service is higher than the supply, then, by advertising their needs,

potential consumers are likely to be more effective than service providers advertising offered services.

One way or another, the potential consumer must acquire sufficient descriptions to evaluate whether a given service matches its needs and, if so, the method for the consumer to interact with the service.

3.2.1.2 Willingness

Associated with all service interactions is intent – it is an intentional act to initiate and to participate in a service interaction. For example, if a service consumer discovers a service via its description in a registry, and the consumer initiates an interaction, if the service provider does not cooperate then there can be no interaction. In some circumstances it is precisely the correct behavior for a service to fail to respond – for example, it is the classic defense against certain denial-of-service attacks.

The extent of a service participant's willingness to engage in service interactions may be the subject of policies. Those policies may be documented in the service description.

Willingness on the part of service providers and consumers to interact is not the same as a willingness to perform requested actions. A service provider that rejects all attempts to cause it to perform some action may still be fully willing and engaged in interacting with the consumer.

3.2.1.3 Reachability

Reachability is the relationship between service participants where they are able to interact; possibly by exchanging information. Reachability is an essential pre-requisite for service interaction – participants MUST be able to communicate with each other.

A service consumer may have the intention of interacting with a service, and may even have all the information needed to communicate with it. However, if the service is not reachable, for example if there is not a communication path between the consumer and provider, then, effectively, the service is not visible to the consumer.

3.2.2 Interacting with services

Interacting with a service involves performing actions against the service. In many cases, this is accomplished by sending and receiving messages, but there are other modes possible that do not involve explicit message transmission. For example, a service interaction may be effected by modifying the state of a shared resource. However, for simplicity, we often refer to message exchange as the primary mode of interaction with a service.

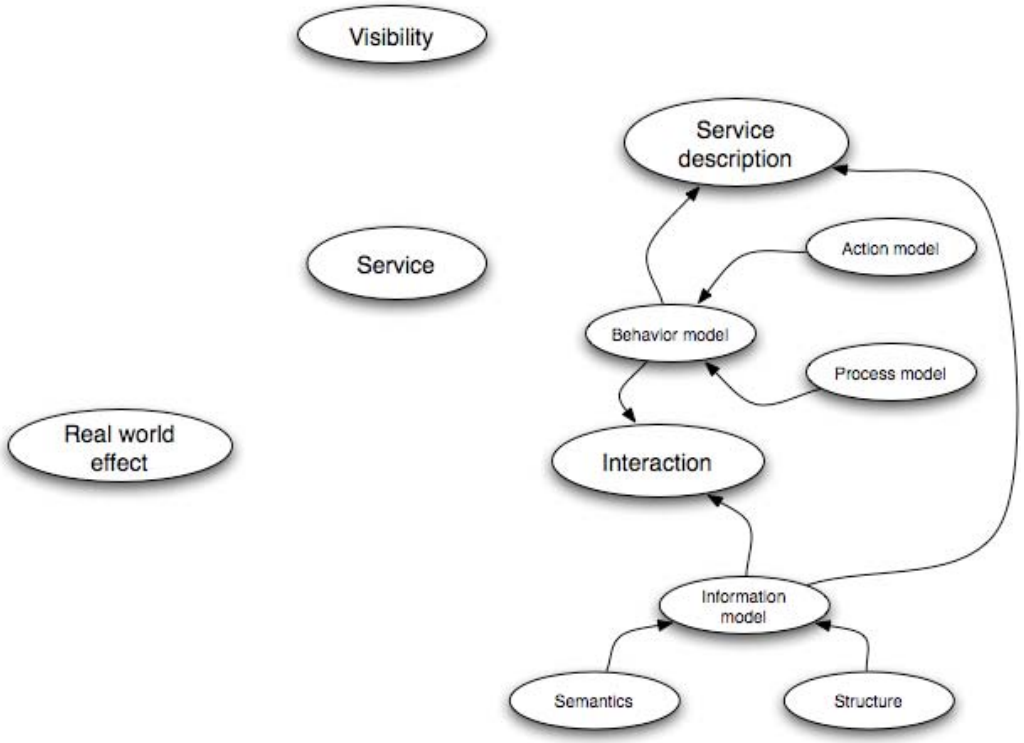


Figure 6 Service Interaction concepts

Figure 6 illustrates the key concepts that are important in understanding what it is involved in interacting with services; these revolve around the service description – which references a **information model** and a **behavior model**.

3.2.2.1 Information model

The information model of a service is a characterization of the information that may be exchanged with the service. Only information and data that are potentially exchanged with a service are generally included within that service's information model.

The scope of the information model includes the format of information that is exchanged, the structural relationships within the exchanged information and also the definition of terms used.

Particularly for information that is exchanged across an ownership boundary, an important aspect of the service information model is the consistent interpretation of strings and other tokens in the information.

The extent to which one system can effectively interpret information from another system is governed by the **semantic engagement** of the various systems. The semantic engagement of a system is a relationship between the system and information it may encounter. This is highly variable and application dependent; for example an encryption service interprets all information as a stream of bytes for it to encrypt or decrypt, whereas a database service would attempt to interpret the same information stream in terms of requests to query and/or modify the database.

Loosely, one might partition the interpretation of an informational block into structure (syntax) and semantics (meaning); although both are part of the information model.

3.2.2.1.1 Structure

Knowing the representation, structure, and form of information required is a key initial step in ensuring effective interactions with a service. There are several levels of such structural information; including the encoding of character data, the format of the data and the structural data types associated with elements of the information.

A described information model typically has a great deal to say about the form of messages. However, knowing the type of information is not sufficient to completely describe the appropriate interpretation of data. For example, within a street address structure, the city name and the street name are typically given the same data type – some variant of the string type. However, city names and street names are not really the same type of thing at all. Distinguishing the correct interpretation of a city name string and a street name string is not possible using type-based techniques – it requires additional information that cannot be expressed purely in terms of the structure of data.

3.2.2.1.2 Semantics

The primary task of any communication infrastructure is to facilitate the exchange of information and the exchange of intent. For example, a purchase order combines two somewhat orthogonal aspects: the description of the items being purchased and the fact that one party intends to purchase those items from another party. Even for exchanges that do not cross any ownership boundaries, exchanges with services have similar aspects.

Especially in the case where the exchanges are across ownership boundaries, a critical issue is the interpretation of the data. This interpretation MUST be consistent between the participants in the service interaction. Consistent interpretation is a stronger requirement than merely type (or structural) consistency – the tokens in the data itself must also have a shared basis.

There is often a huge potential for variability in representing street addresses. For example, an address in San Francisco, California may have variations in the way the city is represented: SF, San Francisco, San Fran, the City by the Bay are all alternate denotations of the same city. For successful exchange of address information, all the participants must have a consistent view of the meaning of the address tokens if address information is to be reliably shared.

The formal descriptions of terms and the relationships between them (e.g., an ontology) provides a firm basis for selecting correct interpretations for elements of information exchanged. For example, an ontology can be used to capture the alternate ways of expressing the name of a city as well as distinguishing a city name from a street name.

Note that, for the most part, it is not expected that service consumers and providers would actually exchange descriptions of terms in their interaction but, rather, would reference existing descriptions – the role of the semantics being a background one – and these references would be included in the service descriptions.

Specific domain semantics are beyond the scope of this reference model; but there is a requirement that the service interface enable providers and consumers to identify unambiguously those definitions that are relevant to their respective domains.

3.2.2.2 Behavior model

The second key requirement for successful interactions with services is knowledge of the actions invoked against the service and the process or temporal aspects of interacting with the service. This is characterized as knowledge of the actions on, responses to, and temporal dependencies between actions on the service.

For example, in a security-controlled access to a database, the actions available to a service consumer include presenting credentials, requesting database updates and reading results of queries. The security may be based on a challenge-response protocol. For example, the initiator presents an initial token of identity, the responder presents a challenge and the initiator responds to the challenge in a way that satisfies the database. Only after the user's credentials have been verified will the actions that relate to database update and query be accepted.

The sequences of actions involved are a critical aspect of the knowledge required for successful use of the secured database.

3.2.2.2.1 Action model

The **action model** of a service is the characterization of the actions that may be invoked against the service. Of course, a great portion of the behavior resulting from an action may be private; however, the expected public view of a service surely includes the implied effects of actions.

For example, in a service managing a bank account, it is not sufficient to know that you need to exchange a given message (with appropriate authentication tokens), in order to use the service. It is also necessary to understand that using the service may actually affect the state of the account (for example, withdrawing cash); that dependencies are involved (for example, a withdrawal request must be less than the account balance); or that the data changes made have different value in different contexts (for example, changing the data in a bank statement is not the same as changing the amount in the account).

3.2.2.2.2 Process Model

The **process model** characterizes the temporal relationships and temporal properties of actions and events associated with interacting with the service.

Note that although the process model is an essential part of this Reference Model, its extent is not completely defined. Some process models MAY include aspects that are not strictly part of SOA – for example, in this Reference Model we do not address the orchestration of multiple services, although orchestration and choreography may be part of the process model. At a minimum, the process model MUST cover the interactions with the service itself.

The reason that orchestration (and choreography) are not part of the SOA RM is that the focus of the RM is on modeling what service is and what key relationships are involved in modeling service.

Beyond the straightforward mechanics of interacting with a service there are other, higher-order, attributes of services' process models that are also often important. These can include whether the service is **idempotent**, whether the service is **long-running** in nature and whether it is important to account for any **transactional** aspects of the service.

3.2.3 Real World Effect

There is always a particular purpose associated with interacting with a service. Conversely, a service provider (and consumer) often has a priori conditions that apply to its interactions. The service consumer is trying to achieve some result by using the service, as is the service provider. At first sight, such a goal can often be expressed as "trying to get the service to do something". This is sometimes known as the "real world effect" of using a service. For example, an airline reservation service can be used to learn about available flights, seating and ultimately to book travel – the desired real world effect being information and a seat on the right flight.

As was discussed in Section 3.1, a real world effect can be the response to a request for information or the change in the state of some defined entities shared by the service participants. In this context, the shared state does not necessarily refer to specific state variables being saved in physical storage but rather represents shared information about the affected entities. So in the example of the airline reservation, the shared state - that there is a seat reserved on a particular flight - represents a common understanding between a future passenger and the airline. The details of actual state changes – whether on the part of the passenger (e.g. fund balances required to pay for the ticket) or of the airline (e.g. that a seat is sold for that flight) - are not shared by the other.

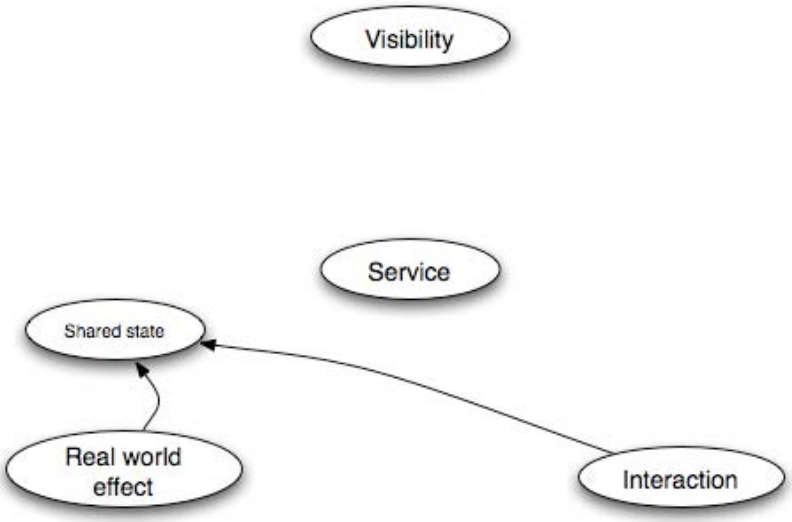


Figure 7 Real World Effect and shared state

In addition, the internal actions that service providers and consumers perform as a result of participation in service interactions are, by definition, private and fundamentally unknowable. By unknowable we mean both that external parties cannot see others' private actions and, furthermore, SHOULD NOT have explicit knowledge of them. Instead we focus on the set of facts shared by the parties – the shared state. Actions by service providers and consumers lead to modifications of this shared state; and the real world effect of a service interaction is the accumulation of the changes in the shared state.

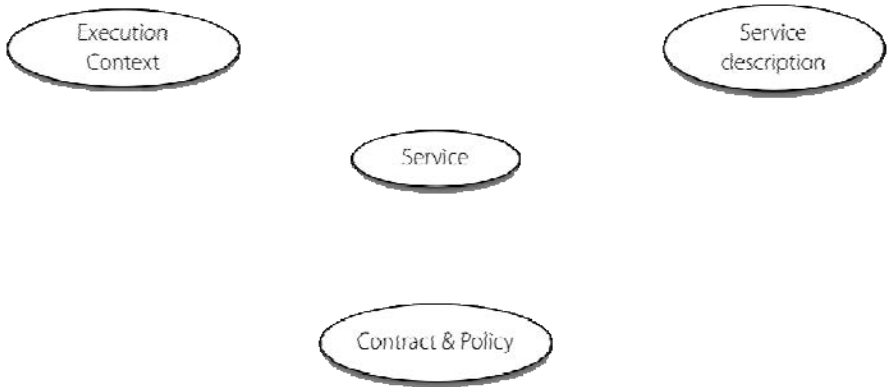
For example, when an airline has confirmed a seat for a passenger on a flight this represents a fact that both the airline and the passenger share – it is part of their shared state. Thus the real world effect of booking the flight is the modification of this shared state – the creation of the fact of the booking. Flowing from the shared facts, the passenger, the airline, and interested third parties may make inferences – for example, when the passenger arrives at the airport the airline confirms the booking and permits the passenger onto the airplane (subject of course to the passenger meeting the other requirements for traveling).

For the airline to know that the seat is confirmed it will likely require some private action to record the reservation. However, a passenger should not have to know the details of the airline internal procedures. Likewise, the airline does not know if the reservation was made by the passenger or someone acting on the passenger's behalf. The passenger's and the airline's understanding of the reservation is independent of how the airline maintains its records or who initiated the action.

There is a strong relationship between the shared state and the interactions that lead up to that state. The elements of the shared state SHOULD be inferable from that prior interaction together with other context as necessary. In particular, it is not required that the state be recorded; although without such recording it may become difficult to audit the interaction at a subsequent time.

3.3 About services

In support of the dynamics of interacting with services are a set of concepts that are about services themselves. These are the service description, the execution context of the service and the contracts and policies that relate to services and service participants.



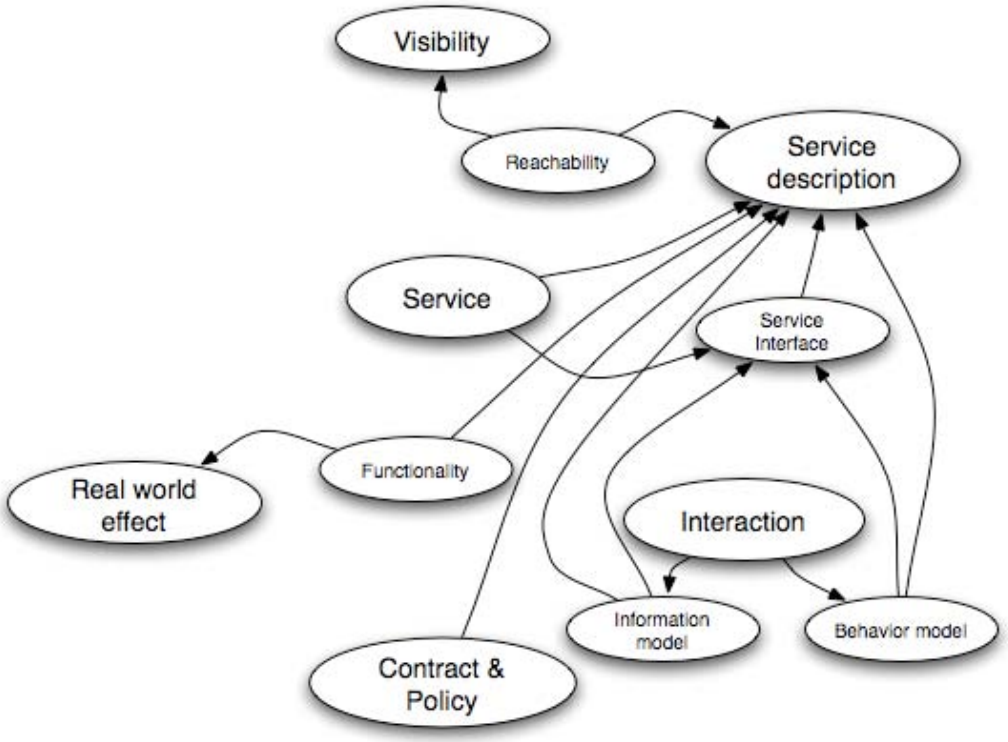
554

555 Figure 8 About services

556 3.3.1 Service description

557 One of the hallmarks of a Service Oriented Architecture is the large amount of associated
558 documentation and description.

559 The service description represents the information needed in order to use a service. In most
560 cases, there is no one “right” description but rather the elements of description required depend
561 on the context and the needs of the parties using the associated entity. While there are certain
562 elements that are likely to be part of any service description, most notably the information model,
563 many elements such as function and policy may vary.



564

565 Figure 9 Service description

566 The purpose of description is to facilitate interaction and visibility, particularly when the
567 participants are in different ownership domains, between participants in service interactions. By
568 providing descriptions, it makes it possible for potential participants to construct systems that use
569 services and even offer compatible services.

For example, descriptions allow participants to discriminate amongst possible choices for service interaction; such as whether the service provides required capabilities, how to access the service, and negotiate over specific service functionality. In addition, descriptions can be used to support the management of services, both from the service provider's perspective and the service consumer's perspective.

Best practice suggests that the service description SHOULD be represented using a standard, referenceable format. Such a format facilitates the use of common processing tools (such as discovery engines) that can capitalize on the service description.

While the concept of a SOA supports use of a service without the service consumer needing to know the details of the service implementation, the service description makes available critical information that a consumer needs in order to decide whether or not to use a service. In particular, a service consumer needs to possess the following items of information:

1. That the service exists and is **reachable**;
2. That the service performs a certain function or set of functions;
3. That the service operates under a specified set of constraints and policies;
4. That the service will (to some implicit or explicit extent) comply with policies as prescribed by the service consumer;
5. How to interact with the service in order to achieve the required objectives, including the format and content of information exchanged between the service and the consumer and the sequences of information exchange that may be expected.

While each of these items SHOULD be represented in any service description, the details can be included through references (links) to external sources and are NOT REQUIRED to be incorporated explicitly. This enables reuse of standard definitions, such as for functionality or policies.

Other sections of this document deal with these aspects of a service, but the following subsections discuss important elements as these relate to the service description itself.

3.3.1.1 Service Reachability

Reachability is an inherently pairwise relationship between service providers and service consumers. However, a service description SHOULD include sufficient data to enable a service consumer and service provider to interact with each other. This MAY include metadata such as the location of the service and what information protocols it supports and requires. It MAY also include dynamic information about the service, such as whether it is currently available.

3.3.1.2 Service Functionality

A service description SHOULD unambiguously express the function(s) of the service and the real world effects (see Section 3.2.3) that result from it being invoked. This portion of the description SHOULD be expressed in a way that is generally understandable by service consumers but able to accommodate a vocabulary that is sufficiently expressive for the domain for which the service provides its functionality. The description of functionality may include, among other possibilities, a textual description intended for human consumption or identifiers or keywords referenced to specific machine-processable definitions. For a full description, it MAY indicate multiple identifiers or keywords from a number of different collections of definitions.

Part of the description of functionality may include underlying technical assumptions that determine the limits of functionality exposed by the service or of the underlying capability. For example, the amounts dispensed by an automated teller machine (ATM) are consistent with the assumption that the user is an individual rather than a business. To use the ATM, the user must not only adhere to the policies and satisfy the constraints of the associated financial institution (see Section 3.3.1.3 for how this relates to service description and Section 3.3.2 for a detailed discussion) but the user is limited to withdrawing certain fixed amounts of cash and a certain number of transactions in a specified period of time. The financial institution, as the underlying

capability, does not have these limits but the service interface as exposed to its customers does, consistent with its assumption of the needs of the intended user. If the assumption is not valid, the user may need to use another service to access the capability.

3.3.1.3 Policies Related to a Service

A service description MAY include support for associating policies with a service and providing necessary information for prospective consumers to evaluate if a service will act in a manner consistent with the consumer's constraints.

3.3.1.4 Service Interface

The service interface is the means for interacting with a service. It includes the specific protocols, commands, and information exchange by which actions are initiated that result in the real world effects as specified through the service functionality portion of the service description.

The specifics of the interface SHOULD be syntactically represented in a standard referenceable format. These prescribe what information needs to be provided to the service in order to access its capabilities and interpret responses. This is often referred to as the service's information model, see Section 3.2.2.1. It should be noted that the particulars of the interface format are beyond the scope of the reference model. However, the existence of interfaces and accessible descriptions of those interfaces are fundamental to the SOA concept.

While this discussion refers to a standard referenceable syntax for service descriptions, it is not specified how the consumer accesses the interface definition nor how the service itself is accessed. However, it is assumed that for a service to be usable, its interface MUST be represented in a format that allows interpretation of the interface information by its consumers.

3.3.1.5 The Limits of Description

There are well-known theoretic limits on the effectiveness of descriptions – it is simply not possible to specify, completely and unambiguously, the precise semantics of and all related information about a service.

There will always be unstated assumptions made by the describer of a service that must be implicitly shared by readers of the description. This applies to machine processable descriptions as well as to human readable descriptions.

Fortunately, complete precision is not necessary – what is required is sufficient scope and precision to support intended use.

Another kind of limit of service descriptions is more straightforward: whenever a repository is searched using any kind of query there is always the potential for *zero or more* responses – no matter how complete the search queries or the available descriptions appear to be. This is inherent in the principles involved in search.

In the case that there is more than one response, this set of responses has to be converted into a single choice. This is a private choice that must be made by the consumer of the search information.

3.3.2 Policies and Contracts

A **policy** represents some constraint or condition on the use, deployment or description of an owned entity as defined by any participant. A **contract**, on the other hand, represents an agreement by two or more parties. Like policies, agreements are also about the conditions of use of a service; they may also constrain the expected real world effects of using a service. The reference model is focused primarily on the concept of policies and contracts as they apply to services. We are not concerned with the form or expressiveness of any language used to express policies and contracts.

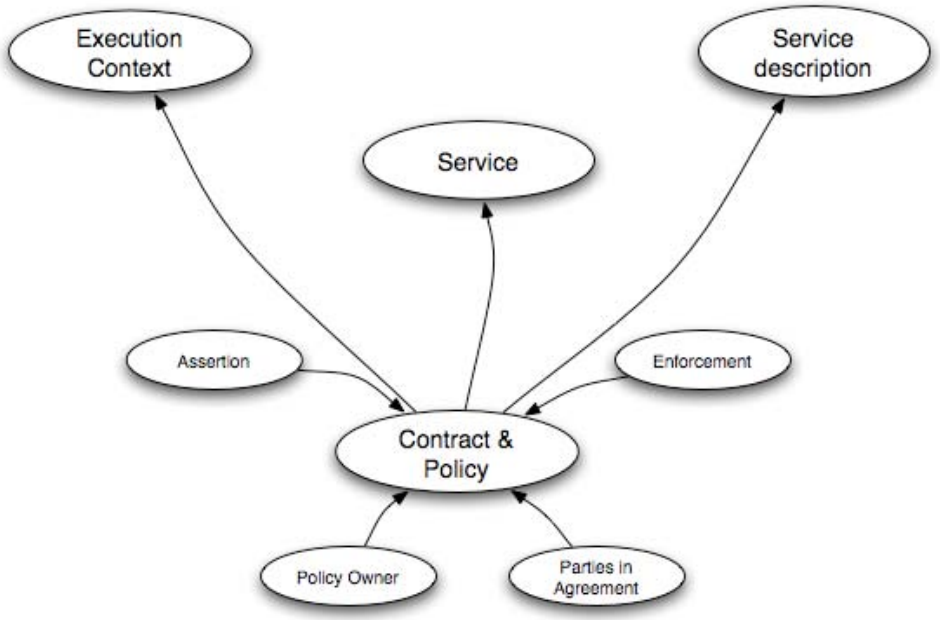


Figure 10 Policies and Contracts

3.3.2.1 Service Policy

Conceptually, there are three aspects of policies: the policy assertion, the policy owner (sometimes referred to as the policy subject) and policy enforcement.

For example, the assertion: “All messages are encrypted” is an assertion regarding the forms of messages. As an assertion, it is measurable: it may be true or false depending on whether the traffic is encrypted or not. Policy assertions are often about the way the service is realized; i.e., they are about the relationship between the service and its execution context, see 3.3.3.

A policy always represents a participant’s point of view. An assertion becomes the policy of a participant when they adopt the assertion as their policy. This linking is normally not part of the assertion itself. For example, if the service consumer declares that “All messages are encrypted”, then that reflects the policy of the service consumer. This policy is one that may be asserted by the service consumer independently of any agreement from the service provider.

Finally, a policy may be enforced. Techniques for the enforcement of policies depend on the nature of the policy. Conceptually, service policy enforcement amounts to ensuring that the policy assertion is consistent with the real world. This might mean preventing unauthorized actions to be performed or states to be entered into; it can also mean initiating compensatory actions when a policy violation has been detected. An unenforceable constraint is not a policy; it would be better described as a wish.

Policies potentially apply to many aspects of SOA: security, privacy, manageability, Quality of Service and so on. Beyond such infrastructure-oriented policies, participants MAY also express business-oriented policies – such as hours of business, return policies and so on.

Policy assertions SHOULD be written in a form that is understandable to, and processable by, the parties to whom the policy is directed. Policies MAY be automatically interpreted, depending on the purpose and applicability of the policy and how it might affect whether a particular service is used or not.

A natural point of contact between service participants and policies associated with the service is in the service description – see Section 3.3.1. It would be natural for the service description to contain references to the policies associated with the service.

3.3.2.2 Service Contract

Whereas a policy is associated with the point of view of individual participants, a contract represents an agreement between two or more participants. Like policies, contracts can cover a wide range of aspects of services: quality of service agreements, interface and choreography agreements and commercial agreements. Note that we are not necessarily referring to legal contracts here.

Thus, following the discussion above, a service contract is a measurable assertion that governs the requirements and expectations of two or more parties. Unlike policy enforcement, which is usually the responsibility of the policy owner, contract enforcement may involve resolving disputes between the parties to the contract. The resolution of such disputes may involve appeals to higher authorities.

Like policies, contracts may be expressed in a form that permits automated interpretation. Where a contract is used to codify the results of a service interaction, it is good practice to represent it in a machine processable form. Among other purposes, this facilitates automatic service composition. Where a contract is used to describe over-arching agreements between service providers and consumers, then the priority is likely to make such contracts readable by people.

Since a contract is inherently the result of agreement by the parties involved, there is a *process* associated with the agreement action. Even in the case of an implicitly agreed upon contract, there is logically an agreement action associated with the contract, even if there is no overt action of agreement. A contract may be arrived at by a mechanism that is not directly part of an SOA – an out of band process. Alternatively, a contract may be arrived at during the course of a service interaction – an in-band process.

3.3.3 Execution context

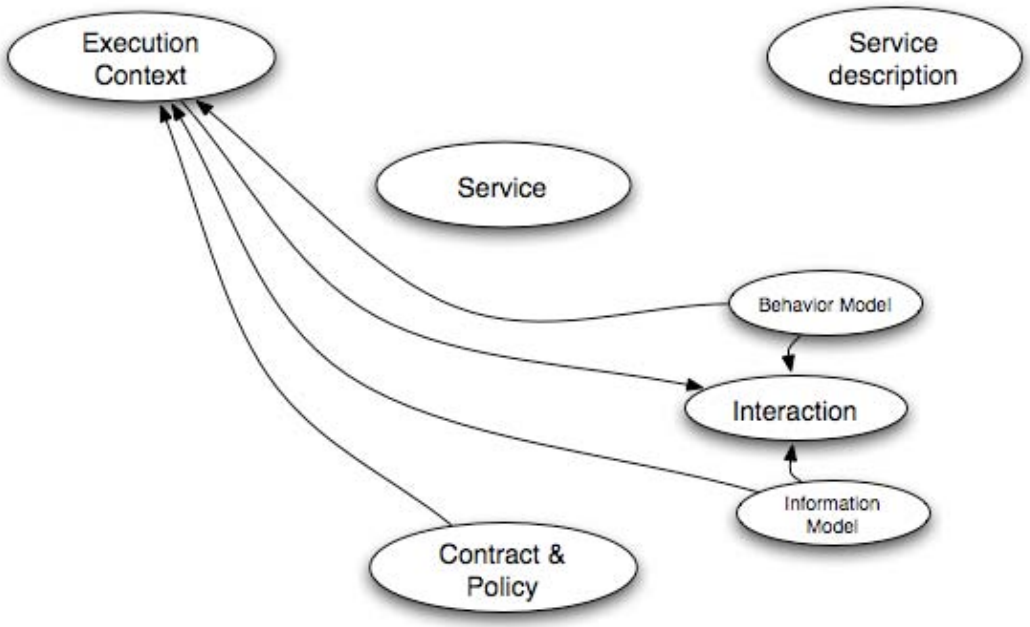


Figure 11 Execution Context

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.

As discussed in previous sections of this document, the service description (and a corresponding description associated with the service consumer and its needs) contains information that can include preferred protocols, semantics, policies and other conditions and assumptions that

describe how a service can and may be used. The participants (providers, consumers, and any third parties as noted below) must agree and acknowledge a consistent set of agreements in order to have a successful service interaction, i.e. realizing the described real world effects. The execution context is the collection of this consistent set of agreements.

The consumer and provider can be envisioned as separate places on a map and, for a service to actually be invoked, a path must be established between those two places. This path is the execution context. As with a path between places, it can be a temporary connection (e.g. a tenuous footbridge of an ad hoc exchange) or a well-defined coordination (e.g. a super highway) that can be easily reused in the future.

The execution context is not limited to one side of the interaction; rather it concerns the totality of the interaction – including the service provider, the service consumer and the common infrastructure needed to mediate the interaction. While there may be third parties, for example, government regulators, who set some of the conditions for the execution context, this merely increases the conditions and constraints needing to be coordinated and may require additional information exchange to complete the execution context.

The execution context is central to many aspects of a service interaction. It defines, for example, a decision point for policy enforcement relating to the service interaction. Note that a policy decision point is not necessarily the same as an enforcement point: an execution context is not by itself something that lends itself to enforcement. On the other hand, any enforcement mechanism of a policy is likely to take into account the particulars of the actual execution context.

The execution context also allows us to distinguish services from one another. Different instances of the same service – denoting interactions between a given service provider and different service consumers for example – are distinguished by virtue of the fact that their execution contexts are different.

Finally, the execution context is also the context in which the interpretation of data that is exchanged takes place. A particular string has a particular meaning in a service interaction in a particular context – the execution context.

An execution context often evolves during a service interaction. The set of infrastructure elements, the policies and agreements that apply to the interaction, may well change during a given service interaction. For example, at an initial point in an interaction, it may be decided by the parties that future communication should be encrypted. As a result the execution context also changes – to incorporate the necessary infrastructure to support the encryption and continue the interaction.

4 Conformance Guidelines

The authors of this reference model envision that architects may wish to declare their work is conformant with this reference model. Conforming to a Reference Model is not generally an easily automatable task – given that the Reference Model's role is primarily to define concepts that are important to SOA rather than to give guidelines for implementing systems.

However, we do expect that any given Service Oriented Architecture will reference the concepts outlined in this specification. As such, we expect that any design for a system that adopts the SOA approach will

- Have entities that can be identified as services as defined by this Reference Model;
- Be able to identify how visibility is established between service providers and consumers;
- Be able to identify how interaction is mediated;
- Be able to identify how the effect of using services is understood;
- Have descriptions associated with services;
- Be able to identify the execution context required to support interaction; and
- It will be possible to identify how policies are handled and how contracts may be modeled and enforced.

It is not appropriate for this specification to identify *best practices* with respect to building SOA-based systems. However, the ease with which the above elements can be identified within a given SOA-based system could have significant impact on the scalability, maintainability and ease of use of the system.

5 References

5.1 Normative

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

5.2 Non-Normative

[W3C WSA] W3C Working Group Note "Web Services Architecture", http://www.w3.org/TR/ws-arch/ , 11 February 2004

A. Glossary

The glossary contains a concise definition of terms used within this specification, but the full description in the text is the normative description.

Action Model

The characterization of the permissible actions that may be invoked against a service. See Section 3.2.2.2.1.

Awareness

A state whereby one party has knowledge of the existence of the other party. Awareness does not imply willingness or reachability. See Section 3.2.1.1.

Behavior Model

The characterization of (and responses to, and temporal dependencies between) the actions on a service. See Section 3.2.2.2.

Capability

A real-world effect that a service provider is able to provide to a service consumer. See Section 2.1.

Execution context

The 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. See Section 3.3.3.

Framework

A set of assumptions, concepts, values, and practices that constitutes a way of viewing the current environment.

Idempotency/Idempotent

A characteristic of a service whereby multiple attempts to change a state will always and only generate a single change of state if the operation has already been successfully completed once. See Section 3.2.2.2.2.

Information model

The characterization of the information that is associated with the use of a service. See Section 3.2.2.1.

Interaction

The activity involved in making using of a capability offered, usually across an ownership boundary, in order to achieve a particular desired real-world effect. See Section 3.2.3.

Offer

An invitation to use the capabilities made available by a service provider in accordance with some set of policies.

Policy

A statement of obligations, constraints or other conditions of use of an owned entity as defined by a participant. See Section 3.3.2.

Process Model

The characterization of the temporal relationships between and temporal properties of actions and events associated with interacting with the service. See Section 3.2.2.2.2.

Reachability

The ability of a service consumer and service provider to interact. Reachability is an aspect of visibility. See Section 3.2.1.3.

Real world effect

The actual result of using a service, rather than merely the capability offered by a service provider. See Section 3.2.3.

Reference Architecture

A reference architecture is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. See Section 1.1.

Reference Model

A reference model is an abstract framework for understanding significant relationships among the entities of some environment that enables the development of specific architectures using consistent standards or specifications supporting that environment.

A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. See Section 1.1.

Semantics

A conceptualization of the implied meaning of information, that requires words and/or symbols within a usage context. See Section 3.2.2.1.2.

Semantic Engagement

The relationship between an agent and a set of information that depends on a particular interpretation of the information. See Section 3.2.2.1.

Service

The means by which the needs of a consumer are brought together with the capabilities of a provider. See Section 3.1.

Service Consumer

An entity which seeks to satisfy a particular need through the use capabilities offered by means of a service.

Service description

The information needed in order to use, or consider using, a service. See Section 3.3.1.

Service Interface

The means by which the underlying capabilities of a service are accessed. See Section 3.3.1.4.

Service Oriented Architecture (SOA)

Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. See Section 2.1.

Service Provider

An entity (person or organization) that offers the use of capabilities by means of a service.

Shared state

The set of facts and commitments that manifest themselves to service participants as a result of interacting with a service.

Software Architecture

The structure or structures of an information system consisting of entities and their externally visible properties, and the relationships among them.

Visibility

The capacity for those with needs and those with capabilities to be able to interact with each other. See Section 3.2.1.

Willingness

A predisposition of service providers and consumers to interact. See Section 3.2.1.2.

B. Acknowledgments

The following individuals were members of the committee during the development of this specification and are gratefully acknowledged:

Participants:

Christopher Bashioum, Mitre Corporation
Prasanta Behera, Individual Member
Kathryn Breininger, The Boeing Company
Rex Brooks, HumanMarkup.org, Inc.
Al Brown, FileNet Corporation
Peter F Brown, Individual Member
Joseph Chiusano, Booz Allen Hamilton
David Ellis, Individual Member
Robert S. Ellinger, Northrop Grumman Corporation
Jeff Estefan, Jet Propulsion Laboratory
Don Flinn, Individual Member
Steve Jones, Capgemini
Gregory Kohring, NEC Europe Ltd.
Ken Laskey, Mitre Corporation
C. Matthew MacKenzie (**secretary**), Adobe Systems
Francis McCabe (**secretary**), Fujitsu Laboratories of America Ltd.
Wesley McGregor, Treasury Board of Canada, Secretariat
Tom Merkle, Lockheed Martin Information Technology
Rebekah Metz, Booz Allen Hamilton
Oleg Mikulinsky, WebLayers, Inc.
Jyoti Namjoshi, Patni Computer Systems Ltd.
Duane Nickull **(chair)**, Adobe Systems
George Ntinolazos, Strata Software Ltd
Joseph Pantella, Individual Member
Ron Schuldt, Lockheed Martin
Michael Stiefel, Reliable Software, Inc.
Danny Thornton, Individual Member
Michal Zaremba, Digital Enterprise Research Institute