Reference Model for Service Oriented Architectures

Working Draft 10, 15 November 2005

Document identifier:

wd-soa-rm-10

Location:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

Editors:

C. Matthew MacKenzie, Adobe Systems Incorporated, mattm@adobe.com
Ken Laskey, MITRE Corporation, klaskey@mitre.org
Francis McCabe, Fujitsu Limited, fgm@fla.fujitsu.com
Peter Brown, Individual, peter@justbrown.net
Rebekah Metz, Booz Allen Hamilton, metz_rebekah@bah.com

Abstract:

This Reference Model for Service Oriented Architectures is an abstract framework for understanding the significant entities and relationships between them within service-oriented systems, and for the development of consistent standards or specifications supporting that environment. It is based on core unifying concepts of SOA and may be used by architects developing specific service oriented architectures or by those needing to explain SOA principles. 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.

While service-orientation may be a popular concept found in a broad variety of applications and domains, we make no attempt to account for the use of the concepts and relationships described in this specification 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.

Table of Contents

1 Introduction

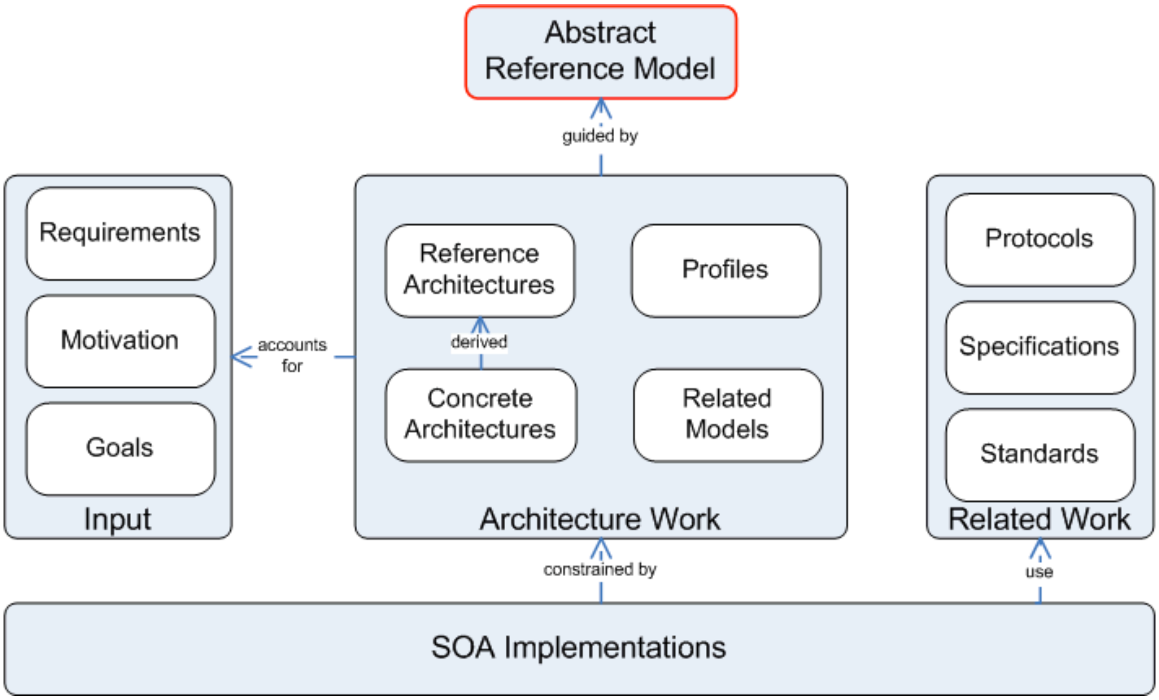
The notion of Service Oriented Architecture (SOA) has received significant attention within the software design and development community. The result of this attention is the proliferation of many conflicting definitions of SOA. Whereas SOA architectural patterns (or *reference architectures*) may be 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 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.

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.

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 for SOA as an abstract and powerful model, irrespective of the various and inevitable technology evolutions that will impact SOA.



1.2 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 architecture.
- Users who need a better understanding of the concepts and benefits of service oriented architecture.

1.3 How to use 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 simplify 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 initially discussing the key concepts behind the reference model and then follow this by more detailed sections on the main concepts. In the first more detailed section, *service* is defined along with *service description*. There then follows a section about interaction between service participants, followed by sections on service policies and expectations. Finally, the concept of service visibility is introduced.

Section 4 addresses compliance with this reference model.

The glossary provides a summary of the definitions made and used within the reference model specification.

1.4 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 Relationships to Other Standards

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

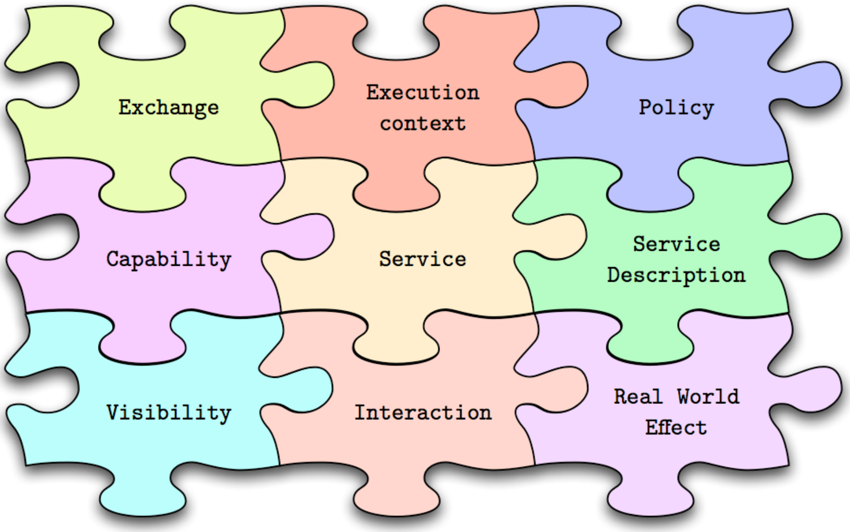
- Considers its work "service oriented";
- Makes (publicly) an adoption statement to use the Reference Model for SOA of this TC as a base or inspiration for their work; and
- Standards or technologies that claim to be service oriented.

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 SOA?

Service Oriented Architecture (SOA) is a paradigm for organizing and using distributed capabilities that may be under the control of different ownership domains. It is natural in such a context 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 to interact. Visibility is typically enhanced through the use of metadata to describe such aspects as functional and technical requirements, related constraints and policies, and mechanisms for interaction. For maximum visibility, metadata must be in a form in which its 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* the 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 together form a path between those with needs and those with capabilities and that permit information to be exchanged, actions to be performed 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 effect**s. At its core, an interaction is "an act" rather than "an object" and the result of an interaction is an effect (or a set/series of effects). We are careful to distinguish *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 (at least) between those involved in the current execution context. Real world effects are, then, manifested in terms of changes to this shared state.

The expected effects, together with relevant preconditions associated with those effects, should be made visible as part of the capability metadata and form an important part of the assessment as to whether a given capability matches similarly described needs. It is not possible to describe every possible effect of using a capability: indeed a cornerstone of SOA is that such knowledge is not necessary.

A concept that is considered central to SOA has not yet been mentioned – that of **service**. Both needs and capabilities exist outside of SOA. What distinguishes SOA is the perceived improvement in bringing needs and capabilities together. **In SOA, services are the mechanism by which needs and capabilities are brought together.** SOA is not the solution of domain problems but rather a way of organizing a wider array of possibilities to generate a domain solution. By itself, SOA does not provide a solution to a difficult domain problem where a satisfactory solution does not already exist. SOA can, however, provide an organizing and delivery paradigm that enables one to get more value from use of both solutions which are locally "owned" and solutions 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 domain solutions.

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 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 invoking the service. In general, entities (people and organizations) offer capabilities through services and act as **service providers**. Those with needs who make use of capabilities through their associated 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 establish whether a consumer satisfies the requirements, if any, of the service provider to be permitted access.

Having described what is SOA, it is appropriate to note several things which are related but are not necessary attributes or restrictions.

- SOA identifies necessary aspects of interactions involving multiple ownership domains; however, it does not directly embody concepts relating to ownership.
- SOA is commonly implemented using Web services, but services can be made visible, support interaction, and generate effects through other implementations.

In most discussions of SOA, the terms "loose coupling" and "coarse-grained" are commonly applied as SOA concepts. However, these terms are subjective and without useful metrics to indicate compliance. In terms of needs and capabilities, SOA is most effective when it focuses on bringing solutions to bear, rather than on "fine-grained" pieces of a particular implementation that may not be reusable beyond a particular solution. Granularity and coarseness are usually relative to detail for the level of the problem being addressed (e.g. one that is more strategic compared with another that considers the issues down to the algorithm level). Counting the number of interfaces or the number or types of information exchanges connected to an interface does not help define the optimum level of detail.

2.2 How is Service Oriented Architecture different?

How does this paradigm of Service Oriented Architecture differ from other approaches to organizing and understanding IT assets? Essentially, there are two areas in which SOA revolutionizes the framework of concepts that functions as a tool for addressing IT solutions.

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 within service descriptions and interfaces.

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.

Unlike Object Oriented Programming paradigms, where the focus is on packaging data with operations, the central focus of SOA is the task or business function – getting something done. This is a more viable basis for large scale systems because it is a better fit to the way human activity itself is managed – by delegation and by trading.

2.3 The Benefits of Service Oriented Architecture

The main drivers for SOA-based architectures are the requirement to facilitate the manageable growth of large-scale enterprise systems, the requirement to facilitate Internet-scale provisioning and use of services and the requirement 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, including 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 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

A service oriented architecture represents a uniform means to offer, discover and interact with capabilities to produce desired effects consistent with measurable preconditions and expectations. This section introduces the main concepts within the SOA paradigm. A detailed discussion of the concepts and their relationships are in the sections that follow.

3.1 Overview of model

A key concept of SOA is that of **service**. 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. SOA is a way to organize the world around this key concept of 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 in the context of SOA, where the capability exists independently of SOA. The term **service** should, therefore, be understood as a set of separate, yet interrelated and more precise concepts. These concepts are an offer, interaction and effect.

The concept of an **offer** follows directly from the dictionary definition of service: 'by one' and 'for another.' In general terms, an offer is a proposal; made by providers which may possess a capability that address a need. In order to use a service, it is necessary to know that it exists, what is accomplished if the service is invoked, how the service is invoked, and other characteristics. Collectively this is the service **visibility**. When given an explicit searchable form, this information allows, for example, prospective consumers to decide if the service is suitable for their current needs and establish whether a consumer satisfies any requirements of the service provider to be permitted access. This information constitutes the **service description**.

The convergence of a capability and a need results in an **interaction**. In an SOA, interaction is effected by exchanging information between service providers and consumers. Typically this is achieved by exchanging messages using a standardized protocol; however, there are many modalities possible for interacting with services.

At its core, an interaction is “an act” rather than “an object.” Therefore, interaction is the focus of the interfaces and behavior necessary to support the interaction. Recall that interaction may, and typically does, involve crossing ownership boundaries. SOA identifies some of the necessary aspects of interactions involving multiple ownership domains; however, it does not directly embody concepts relating to ownership.

The final key concept is the **real world effect** of using services; it is always the case that there is an intended purpose to providing a service and similarly to using a service.
Given that there is often an ownership boundary between the service provider and consumer, there is a natural distinction to be drawn between the public interactions with a service and the private actions of both the service provider and consumer. This distinction maintains and encourages independence of each service participant which, in turn, greatly enhances the scalability and security attributes of SOA. Focus can be directed to the public aspects of using a service by examining the **conditions** of using a service and the **expectations** that arise as a result of using the service. Service conditions are loosely associated with the **service policies** and the expectations with **service contracts**.

3.2 The Reference Model

3.2.1 Service

A **service** is a means to 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 one 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 invoked through a service interface (see Section 3.2.2.4), 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 data model exposed through the published service interface and (2) any information included as metadata to describe aspects of the service which are needed by service consumers to determine whether a given service is appropriate for the consumer's needs. The consequence of invoking a service is a realization of one or more real world effects. The effects may include:

1. information returned in response to a request,
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), the service consumer does not typically know how the state change is effected. In either case, the service consumer would need to provide input parameters required by the service and the service would return information, status indicators, or error descriptions, where both the input and output are as described by the data model exposed through the published service interface. Note that the service may be invoked without requiring information from the consumer (other than a command to initiate action) and may accomplish its functions without providing any return or feedback to the consumer.

The service concept above emphasizes a distinction between a capability that represents some functionality created to address a need and the point of access to bring that capability to bear in the context of SOA. It is assumed that capabilities exist outside of the 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.

3.2.2 Service description

The service description represents the information needed in order to use a service. It may be considered part of or the complete set of the metadata (see Section 3.2.3) associated with a service. In any case, the service description overlaps and shares many common properties with service metadata. In most cases, there is no one "right" set of metadata but rather the metadata content depends on the context and the needs of the parties using the associated entity. The same holds for a service description. While there are certain elements that are likely to be part of any service description, most notably the data model, many elements such as function and policy may vary.

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, in turn, 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 must possess the following items of information:

1. That the service exists and is **reachable** (i.e., the service is **visible** to the service consumer and there are sufficient mechanisms in place for the service participants to be able to interact);
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.

Subsequent sections of this document will deal with these aspects of a service in detail but the following subsections will describe the relationship of these information items to the service description.

3.2.2.1 Service Reachability

A service description should include sufficient data to permit a service consumer and service provider to exchange information. This might include metadata (such as the location of the service and what information protocols it supports and requires) and information that allows the service consumer to determine if the service is currently reachable or not.

3.2.2.2 Service Functionality

Item 2 relates to the need to unambiguously express the function(s) of the service and the real world effects (see Section **Error! Reference source not found.**) that result from it being invoked. This portion of the description needs to 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-process-able definitions. For a full description, it may be useful to 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 3.2.2.3 for how this relates to service description and Section **Error! Reference source not found.** 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.

3.2.2.3 Policies Related to a Service

Items 3 and 4 from Section 2.2.2 relate to the service description's support for associating constraints and 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 and policies.

In some situations the consumer may similarly provide an indication of its constraints and policies to support a service’s need to do a similar evaluation of suitability. Thus, both prospective consumers and providers are likely to use the service description to establish what Section 3.3.3 refers to as the **execution context**.

3.2.2.4 Service Interface

The service interface is the means referred to in Item 5 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 exercise its functionality and/or the results of the service invocation to be returned to the service consumer. This logical expression of the set of information items associated with the consumption of the service is often referred to as the service’s data model. It should be noted that the particulars of the standard reference-able format is beyond the scope of the reference model. However, requiring that mechanisms be available (in order to define and retrieve such definitions) is fundamental to the SOA concept.

3.2.2.5 An Example of Using Information Contained in the Service Description

The following example may help clarify the concepts related to service and service description.

A utility has the capacity to generate and distribute electricity (the underlying capability). A consumer accesses electricity generated (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, which voltage is used and possible limits to the load(service description). The utility presumes that the customer will only connect devices that are compatible with the voltage provided; and the consumer in turn assumes that compatible devices can be connected without damage or harm (service assumptions).

A residential or business user will need to open an account with the utility in order to use the supply (service contract) and the utility will meter usage and expects the consumer to pay for use at the rate prescribed (service contract). Provided that the consumer utilizes the correct plugs and does not overload the system (service policy), the consumer can receive electricity using the service.

Another person (say, 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 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). In this example, the underlying capability would still exist and be usable even if every device were required to be hard-wired to the utility's equipment, but this would result in a very different service and service interface.

3.2.3 Descriptions and Metadata

One of the hallmarks of a Service Oriented Architecture is the degree of documentation and description associated with it; particularly *machine process-able descriptions* – otherwise known as *metadata.*

The purpose of this metadata is to facilitate integration, particularly across ownership domains. By providing public descriptions, it makes it possible for potential participants to construct applications that use services and even offer compatible services. Standardizing the formats of such metadata reduces the cost and burden of producing the descriptions necessary to promote reuse and integration.

3.2.3.1 The roles of description

An important additional benefit of metadata – as opposed to informal natural language descriptions – is its potential to facilitate automated software development. Both service providers and service consumers can benefit from such automation – reducing the cost of developing such systems.

For example, metadata can be used as a basis of discovery in dynamic systems. Metadata can assist in managing a service, validating and auditing usage of services which may also be simplified by rich metadata. It can also help ensure that requirements and expectations (regarding the content of any data interchanged) are properly interpreted and fulfilled.

3.2.3.2 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 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 either – what is required is sufficient precision to enable required functionality.

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.

There may be many reasons why a multiplicity of responses is returned: there might be several versions of the service, there might be competing services that offer overlapping functionality and there might be services from multiple different providers.

In the case that there is more than one response, this set of responses has to be converted into a choice of a single service in order for a service consumer to ensure the required function performed. In a multi-provider scenario, that choice must also take into account real world aspects of the service – such as whether the service consumer can identify the provider, can or should trust the provider, and whether the provider is reliable and timely in delivering the service offered. It is unlikely that all such factors can be easily and securely encoded in descriptions and search criteria.

3.3 Interacting with services

Interacting with a service involves exchanging information with the service and 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. However, for simplicity, we often refer to message exchange as the primary mode of interaction with a service. The forms of information exchanged and understood, together with the mechanisms used to exchange information, constitute the **service interface** – see Section 3.2.2.4.

The key concepts that are important in understanding what it is involved in interacting with services are the **data model**, the **process model,** the **execution context** and the **expectations** about the interaction.

3.3.1 Data model

The data model of a service is a characterization of the information associated with the use of the service.

The scope of the data model includes the format of exchanged information, the structural relationships within those documents and the definition of terms used. Typically, only information about, and data potentially included in, an exchange with a service are generally considered as being part of that service's data model.

There are two important aspects of a data model that are important in interpreting information exchange – the structure of the information and the meaning assigned to elements of the information. Particularly for information that is exchanged across an ownership boundary, the interpretation of strings and other tokens in the information is a critical part of the semantics of the interaction.

3.3.1.1 Structure

Understanding the representation, structure and form of information exchanged is a key initial step in ensuring effective interactions with a service. There are several levels of such structural information; ranging from the encoding of character data, through the use of formats such as XML, SOAP and schema-based representations.

A described data model typically has a great deal to say about the form of messages, about the types of the various components of messages and so on. However, pure “typed” information is not sufficient to completely describe the appropriate interpretation of data.

3.3.1.2 Semantics and Ontology

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 if for exchanges that do not cross any ownership boundaries, exchanges with services have similar aspects: this is an update to the customer profile with these changes.

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.

For example, 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.

An ontology is a formal description of terms and the relationships between them in a given context. It will include information about how terms should be interpreted within a given context, constraints on and functions of valid values for the data and associated properties, as well as information about possible relationships of some terms to other terms (hierarchical, class/sub-class, associative, dependent, etc.).

The role of explicit ontologies in an SOA is to provide 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.

Ontologies also provide a point of context to facilitate the *reinterpretation* of data – for example that a 3/8” steel washer may be a potential replacement for a 1cm spacer. Such a reinterpretation is effectively represented as a particular traversal of the graph of concepts and relationships embodied in the ontology. How much automation of ontology walking is appropriate will depend on the nature of the service and the service participants.

Note that, for the most part, it is not expected that service consumers and providers would actually exchange ontologies in their interaction – the role of the ontology is a background one – it facilitates sound interactions. Hence ontology references are mostly to be found in **service descriptions**.

More specifically, and in order for a service to be consistent, the service should make consistent use of terms as defined in an ontology. 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.

3.3.2 Behavioral model

The second key requirement for successful interactions with services is knowledge of the process or temporal aspects of interacting with the service. Loosely, this can be 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 service, the actions available to a service consumer might 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 service. Only after the user's credentials have been verified will any action that queries and/or updates the database be accepted. The sequences of actions involved are a critical aspect of the knowledge required for successful use of the secured database service.

There are other aspects of the behavior of services that are important. These include, for example, whether the service is transactional, idempotent or long running. As a particular example, a service that supports updating an account balance with a transaction is typically idempotent; i.e., the state of the account would not be affected should a subsequent interaction be attempted for the same transaction.

3.3.2.1 Action model

The **action** model of a service is about the individual 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 follow not precede an authentication); 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 actual data representing the amount in an account).

3.3.2.2 Process Model

The **process model** characterizes the temporal relationships between actions and events associated with interacting with the service. It is fairly common to partition the process model associated with a service into two levels: the particular sequences of operations needed to achieve single service exchanges and longer term transactions. These two levels may be nested – a long running transaction is often composed of sequences of exchange patterns.

Note that although the process model is an essential part of this Reference Model, its extent is not completely defined. In some architectures the process model will 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 of a given architecture. At a minimum, the process model must cover the interactions with the service itself.

3.3.2.3 Higher-order attributes of processes

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.

A service is idempotent if subsequent attempts to perform identical transactions are discounted. For example, it often important that a bank will only cash a check once – subsequent attempts to cash the same check should be ignored, rejected or initiate an alert process. Note that idempotency is not the same as effect-free or stateless: a service that always returns the same results is idempotent, but only by virtue of the fact that it does not change from invocation to invocation.

Idempotency is an important attribute of a service in an environment where there is a significant possibility that the interaction between the service provider and consumer may be interrupted – whether by a network issue or simply one of the parties dropping out. A strategy for recovering from such a breakdown is to attempt to repeat the interaction – an idempotent service is required to ignore such repetitions should the transaction have been completed beforehand.

A service is long-running if the activities engendered by an interaction are likely to persist beyond the immediate interaction itself. For example, a classic book selling service might be viewed as a long-running service: the activity started by the purchase of the book may take days or weeks to complete. It can be important to account for a long-running process as it has implications for the kinds of infrastructure needed – both by the service provider and by the service consumer – in order to be able to keep track of the progress of the interaction.

Often, once a business-level contract has been agreed on, it can be difficult or impossible to simply cancel the consequences of the agreement. This is particularly an issue when the agreement of several parties is necessary simultaneously. For example, booking a vacation may require a flight ticket as well as a hotel room – without either component the result is not a vacation. However, the airline typically will not have a relationship with the hotel. If there are no hotel rooms available for the proposed vacation then the airline ticket will need to be canceled.

The process of reversing a previously completed transaction – backing out of the airline booking for example – is likely to be quite different to the process for the original transaction; possibly even involving a different service. Knowledge of such compensatory actions is a key aspect of interacting with transactional services.

3.3.3 Actualized Services

The **execution context** of a service interaction is the set of infrastructure elements, process entities and policy assertions that are deployed as part of the instantiated service interaction. In effect, the execution context defines the point of contact between abstractions such as service descriptions which are mostly about the potential for interaction and an actually executing service. It is the point of measurement between the service description and reality, between theory and practice.

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.

The execution context is central to many aspects of a service interaction. It defines, for example, the decision point for any 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 service interaction.

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 – it is where the *symbol grounding* happens as it were. A particular string has a particular meaning in a service interaction in a particular context – the execution context.

3.4 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 interacting with 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 in order to book travel – the desired real world effect being a seat on the right airplane.

The internal actions that a service providers and consumers perform as a result of participation in service interactions are, by definition, private and fundamentally unknowable.[1] 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 state that is shared between the parties – the **shared state**. Actions by service providers and consumers lead to modifications of this shared state; and that in turn leads to modified **expectations** by the participants.

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, both 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. By minimizing assumptions about how the airline fulfils its contracts, the potential for smooth interoperation is maximized. Such minimization principles represent a key success factor for scalability.

Note that there does not need to be a third party to act as a kind of *escrow* for the shared state between service providers and consumers. The elements of the shared state are inferred from the communication that has occurred between the participants together with other context as necessary. Of course, in the case where adjudication is a possibility, it becomes prudent to *record* the interaction – so that disputes can be arbitrated.

Although there is not necessarily a one-to-one correspondence, the natural container for the conditions applying to a service is the **service policy**. Similarly, the natural container for the expectations arising from a service is the **service contract**.

3.5 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.

[1] A similar analysis applies to service consumers: just how a consumer of a service decides which requests to make and which actions to perform is something that the service provider cannot determine.

3.5.1 Service Policy

A policy is a statement of the obligations, constraints or other conditions of use of a given service that expresses intent on the part of a participant. More particularly, policies are a way for expressing the relationship between the **execution context** and the **data** and **behavior models** associated with the service.

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 triple-DES 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.

A policy always represents a participant’s point of view. An assertion becomes the policy of a participant when they make it their policy. This linking is normally not part of the assertion itself. For example, if the service consumer declares that “All messages are triple-DES 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 need to be automatically interpreted, depending on the purpose and applicability of the policy and whether 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 3.2.2. It would be natural for the service description to contain references to the policies associated with the service.

3.5.2 Service Contract

Where 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.

Thus, following the analysis 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. This facilitates automatic service composition, for example. 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.

3.6 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 to interact.

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 – typically the initiator in a service interaction must be aware of the other parties – willingness – the parties must be predisposed to interactions – and ability – the participants must be able to exchange information as part of a service interaction.

3.6.1 Awareness

A key aspect of visibility is 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 of service offerings is often mediated by various *discovery* mechanisms. For a service consumer (say) to discover a service provider, 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 finding that information.

Service discoverability 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, an awareness of the existence and capabilities of the service can become known to potential consumers. The extent to which the discovery 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 then service providers advertising offered services.

One way or another, the potential consumer must acquire a sufficient description to evaluate whether the service matches their expectations and, if so, the method for the consumer to establish a contract and invoke the service.

3.6.2 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.

Of course, 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.

3.6.3 Reachability

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 communication path between the consumer and provider, then, effectively, the service is not visible to the consumer.

Reachability is the relationship between service participants where they are able to exchange information as part of service interactions. Reachability is closely connected to the concept of **execution context** (see Section 3.3.3) – an important requirement for an execution context is to establish that service participants can communicate with each other.

4 Conformance Guidelines

The authors of this reference model envision that architects may wish to declare their architecture 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,
- Such entities will have descriptions associated with them,
- Service entities will have identifiable interaction models, including models of the information exchanged by the services and the temporal behavior of the services
- It should be possible to identify a means by which consumers of services and providers of services are able to engage; and
- That there will be identifiable aspects of service entities that correspond to the policies relating to the conditions of use of services and to the expectations that result from interacting with services.

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

Appendix A. Glossary

EDITOR'S NOTE TO THE READER: This section is currently in flux. Please do not submit comments/issues on/against this appendix.

Terms that are used within this Reference Model are often also found in other specifications. In order to avoid potential ambiguity, this glossary locally scopes the definitions of those terms for the purpose of this Reference Model and thus overrides any other definitions.

Action Model

> The characterization of the permissible actions that may be invoked against a service.

Addressability

> A state of knowledge of a participant whereby information exists that could, in principle, permit a participant to interact with the addressable party. Addressability does not imply reachability.

Awareness

> A state whereby one party has knowledge of the existence of the other party. Awareness does not imply addressability or reachability.

Architecture

> A set of artifacts (that is: principles, guidelines, policies, models, standards and processes) and the relationships between these artifacts, that guide the selection, creation, and implementation of solutions aligned with business goals.
>
> Software architecture is the structure or structures of an information system consisting of entities and their externally visible properties, and the relationships among them.

Authentication

> The act by which one entity establishes – to an agreed level of confidence – the identity of another.

Awareness

> Information that leads a service provider and/or consumer to be able to act on knowledge of the other's existence.

Behavioral Model

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

Capability

> A real-world effect that a service provider is able to provide to a service consumer.

(Service) Consumer

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

Contract

The agreement between a service provider and a consumer, often including conditions of use of a service and an indication of the expected real world effect.

Data Model

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

Discoverability

The possibility that service consumers and service providers can be brought together, and the mechanisms by which this is achieved.

Execution context

A set of technical and business elements that permit information to be exchanged and actions to be performed when the “theory” of a service description, policies and contract become the “practice” of an actual running service.

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 been already been successfully completed once.

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.

Interface

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

Message

A serialized set of data that is used to convey information and/or actions from one party to another.

Metadata

A set of properties of a given entity which are intended to describe and/or indicate the nature and characteristics of the entity.

Offer

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

Ontology

A formal description of terms and the relationships between them in a given context.

Opaqueness

The extent to which an agent is able to interact successfully with a service without detecting how the service is implemented.

Policy

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

Process Model

The characterization of the temporal relationships between actions and events associated with interacting with a service.

(Service) Provider

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

Reachability

The state is which a service is visible to potential consumers and capable of being interacted with.

Real world effect

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

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 is based on a small number of unifying concepts. A reference model is not directly tied to any standards, technologies or other concrete implementation details, but it does seek to provide a common semantics that can be used unambiguously across and between different implementations.

Semantics

A conceptualization of the implied meaning of information, shared between the service consumer and the service provider, that requires words and/or symbols within a usage context.

Service

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

Service description

A set of information describing a service, sufficient to allow a potential consumer to ascertain, where appropriate:

- the identity of (and/or information about) the service provider;
- the policies, parameters and terms of use of the service;
- of the information necessary to interact with the service;
- what is accomplished when the service is invoked,
- and thus be able to use the service as intended by the provider.

Service Oriented Architecture (SOA)

A software architecture of services, policies, practices and frameworks in which components can be reused and repurposed rapidly in order to achieve shared and new functionality. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

Visibility

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

Appendix B. Acknowledgments

The following individuals were members of the committee during the development of this specification:

[TODO: insert cte. Members]

Appendix C. 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.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.