Reference Model for Service Oriented Architectures

Working Draft 08B, 16 September 2005

Document identifier:
wd-soa-rm-08B

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 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 explanation SOA. 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.

While service-orientation may be a popular concept found in a broad variety of applications, this reference model scopes itself to the field of software architecture.

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

72

1 Introduction

1.1 What is a reference model

A reference model consists of a set of clearly defined concepts, axioms and relationships within a particular problem domain, independently of specific implementations, conventions, activities or organizations. The purpose of a reference model is to facilitate the design of systems, to establish a common set of terminology as it applies to the domain and to encourage best practice where possible.

In the field of information technology, specific *architectures* may be developed to promote a common approach to solving particular problems. A group of such architectures – Service Oriented Architectures or SOA.

SOA have received significant attention within the software design and development industry in recent times resulting in many conflicting definitions of “service-oriented architecture”. Whereas SOA architectural patterns (or *reference architectures*) may be developed to explain and underpin the 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 *any* SOA. A reference model is a foundational work that can and should be used to develop architectural patterns and promote effective discourse on derived works.

The goal of this reference model document is to define the essence of the service oriented architecture paradigm, and emerge with a vocabulary and a common understanding of SOA. It should provide a normative reference that remains relevant for SOA as an abstract and powerful *model*, irrespective of the various and inevitable technology evolutions that we experience in this industry and that will impact on specific SOA *implementations*.

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 / analysts developing specifications that relate to or make use of the service-oriented paradigm.
- Chief Information Officers and other decision makers seeking a "consistent and common" understanding 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 promote rapid understanding.

This first 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 of reference models and their use.

Section 2 introduces the Reference Model for SOA. First, the main axioms, key concepts and relationships between those concepts are introduced followed by more detailed sections on the main concepts: a *service* is defined along with *service description*. There then follows a section detailing interaction between services, followed by service policies and expectations. Finally, the concept of service discoverability is introduced.

This section is provided for the benefit of multiple audiences:

- Non-technical readers may use this section to gain an explicit understanding of the core principles of SOA.
- Architects are encouraged to use this section as guidance for developing specific service oriented architectures. Section 2 and its subsections are designed to provide guidance for consistent logical divisions of components within architectures. It also helps architects adhere to the basic principles of service-oriented design.

Section 3 addresses what it might mean to be conformant with this reference model.

The glossary provides definitions of terms which are relied upon within the reference model specification but do not necessarily form part of the specification itself.

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]**.

Comment [m1]: Document does not consistently follow conventions, we need to fix. There are words bolded and color coded with no explanation as to why they are.

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"; and/or
- Makes (publicly) an adoption statement to use the Reference Model for SOAa of this TC as a base or inspiration for their work.

Additionally, there are a large number of standards and technologies that are related by the fact they claim to be or are "service oriented".

Any work that aligns with the functional areas of SOA such as the service, service description, advertising mechanism, service data model or service contract are likely to be directly related.

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

2 The Reference Model

This reference model for service-oriented architectures describes concepts and relationships that are fundamental in describing SOA architecture patterns (i.e. SOA reference architectures) and specific SOA architectures applied to the solution of specific problems. In general, a service-oriented architecture represents a uniform means to discover and access distributed services that invoke functionality and produce desired effects with measurable preconditions and expectations. Such services hide implementation details but have associated service descriptions to provide sufficient information to understand the technical and business requirements for invoking the service. The actual decision (or agreement) to invoke a service is often contingent on understanding and complying with those requirements.

While such a description of SOA gives a flavor for why it is of interest, it is not sufficient for understanding the primary SOA concepts that must be utilized in designing a SOA and effectively using an SOA. The remainder of this section introduces the main concepts and a detailed discussion of the concepts and their relationships are in the sections that follow.

2.1 Overview of model

A key concept of SOA is that of a **service**. In general, people and organizations create capabilities to solve or support the solution for problems they face in the course of their business. SOA is conceived as a way of making those capabilities visible and supporting standard means of access so the existing capabilities can be repurposed or new capabilities can be readily substituted to improve the solutions. A service is a means to access such capabilities.

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, both to allow prospective consumers to decide if the service is suitable for their current needs and to establish whether a consumer satisfies any requirements of the service provider to be permitted access. Such information constitutes the **service description**.

Services are accessed in order to achieve particular effects. However, the nature of SOA are such that there is an “arm’s length” relationship between service providers and consumers. As a

result, there is a distinction to be drawn between the public interactions with a service and the private actions of the service provider and consumer. A great strength of the scalability and security attributes of SOA is that this distinction maintains and encourages independence of each service participant. We can focus on 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. We loosely associate the service conditions with the **service policies** and the expectations with **service contracts**.

Another key concept in SOA is that of **service interaction**. Although services are accessed in order to achieve particular desired effects, this 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 using services.

Finally we identify **discoverability** as a key concept of SOA. Discoverability refers to the possibility and mechanisms by which service consumers and providers can be brought together. There are many possible mechanisms by which discoverability may be achieved: registries or repositories of service descriptions are undoubtedly powerful means of achieving this, but SOA is not limited to these.

2.2 The Reference Model in Detail

2.2.1 Service

A service is a mechanism to enable access to a set of 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 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, 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 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 an expectation of one or more real world effects. The effects may include:

1. information returned in response to a request for that information, including information returned as a result of prior interactions with the service,
2. processing done in response to a request to change the state of defined entities, or
3. some combination of (1) and (2).

Note, the user 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 user does not typically know how the state change is effected. In either case, the service consumer would need to provide input parameters defined (either required or optional) 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 input 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 outline above of the service concept has emphasized a distinction between a capability that represents some functionality created to address a problem or a need; and the service that forms the point of access to bring that capability to bear in the context of SOA. It is assumed that the capability was created and exists outside of the SOA and one of the major benefits of SOA is enabling the capability so as to be applicable to an expanded realm of relevant problems. 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 is terms of a clear expression of the nature of SOA and the value it provides.

2.2.2 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 associated with a service (see Section 2.2.3) but 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. However, the service description should be represented through use of a standard, reference-able format that can accommodate the necessary variations and lend themselves to common processing tools (such as discovery engines) that can make use of the service description.

Comment [m2]: What does this mean?

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 to use a service how. In particular, a service consumer must possess the following items of information:

1. That the service exists and is available;
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.

Comment [KL3]: We've discussed this before but in writing this section it seems that an important concept is not only that there is a service and it's the service I want to use, and I know how to invoke the service, but also some mechanism that lets me know the service is available to be invoked or it can't. It is an implementation decision whether to inform the consumer in advance (or for the consumer to be able to check) about availability or to generate a fault if the service is not accessible. But something on availability/presence seems to be needed.

Comment [FGM]: Reworded away from invoke to interaction.

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

2.2.2.1 Service Availability

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 whether the service is currently available or not.

2.2.2.2 Service Functionality

Item 2 relates to the need to unambiguously express the function(s) of the service and the real world effects (see section 2.4) 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 sections 2.2.2 for how this relates to service description and section 2.4 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.

2.2.2.3 Policies Related to a Service

Items 3 and 4 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 (and the consumer description) to mutually establish what section 2.3.3 refers to as the *execution context*.

Comment [FGM: This is the ifirst that we have heard of the consumer description!

Comment [FGM: Propose deleting.

Comment [FGM: Deleted as unnecessary. Shoot me.

2.2.2.4 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 reference-able 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 requester. 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.

While this discussion refers to a standard reference-able syntax for service descriptions, we do not specify 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.

Comment [FGM: You can shoot me here too.

2.2.2.5 An Example of Using Information Contained in the Service Description

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

> A utility has the capacity to generate and distribute electricity (the service resource). 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; 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 constraint) and the utility will meter usage and expect the consumer to pay for use at the rate prescribed (service contract). 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 resource 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. In short, it would not be service-oriented.

2.2.3 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 with minimal human-level contact between them.

2.2.3.1 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, it can assist in managing a service, validating and auditing uses of services may also be simplified by rich metadata. It can help ensure that requirements and expectations (regarding the content of any data interchanged) are properly interpreted and fulfilled.

2.2.3.2 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 process-able metadata as well as to human readable documentation.

Comment [m9]: I removed "(This is Gödel's incompleteness result in another guise.)", because I agree with Peter that it doesn't need to be there…it seems a bit too debatable, and came off a lot like a flashy display of erudition. It was originally placed right after this sentence.

Luckily, complete precision is not necessary either – what is required is sufficient precision to enable required functionality.

Another kind of limit of descriptions is more straightforward: describing a service (for example) does not eliminate the requirement for making a choice. For example, a service directory might have the descriptions of many services – provided by many organizations. An automatic search of that directory is therefore likely to return multiple responses to any mechanical search criteria. At some point this set of responses has to be converted into a choice of a single service in order for a service consumer (say) to see the required function performed. In a multi-vendor scenario, that choice must also take into account real world aspects of the service – such 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 such factors can be easily and securely encoded in descriptions and search criteria.

2.3 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 to and from the service end-point; but there are other modes possible that do not involve explicit message sending. However, for simplicity, we often refer to message exchange as the primary mode of interaction with a service. Together the types of information exchanged and actions performed constitute the **service interface** – see section 2.2.2.

There are three key concepts that are important in understanding what it is involved in interacting with services – the **data model**, the **process model** and the **execution context**.

Comment [AE10]: Didn't we decide to call this the "behavioral aspects" or is the behavioral aspect simply a sub-aspect of it?

Comment [FGM: Process model seems more accurate

2.3.1 Data model

The data model of a service is a characterization of the information that may be exchanged with the service.

The scope of the data model includes the format of documents and messages, the structural relationships within those documents and also the ontology of terms used within those documents.

The information and data that might be stored or internally manipulation by a service are not generally included within that service's data model. That is part of the service implementation.

2.3.1.1 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; ranging from the encoding of character data, through the use of formats such as XML, SOAP and schema-based representations.

2.3.1.2 Ontology

Particularly for messages, an important aspect of the service data model is the interpretation of strings and other tokens in the data. Loosely, one might partition the interpretation of a message into structure (syntax) and meaning (semantics); although both are part of the data model.

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. 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. For example, a street can be found in a city, but not vice versa.

An ontology is a formal description of terms and the relationships between them in a given context. Most commonly, the relationships are class relationships – one term represents a concept that is a sub-class of another. However, relationships are not limited to the sub-class relationships; other aspects of concepts can also be usefully represented; such as the range of possible values given property can take and whether the property is functional or not.

The role of explicit ontologies is to provide a firm basis for selecting correct interpretations for tokens in messages. For example, in the address example above, an ontology can be used to capture the appropriate distinction between street name and city name; so much so that in many

cases it is possible to automatically map the contained information from one representation to another.

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.

2.3.2 Behavior model

The second key requirement for successful interactions with services is knowledge of the behavioral or process aspects of 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 news subscription service, a successful use of the service involves initially registering a subscription with the service; which will then be followed by an irregular series of one-way notifications of news items. Key to the use of such a service is the knowledge that you must first register your preferences and that you will then receive messages without further prompting.

Another example is a service that supports updating an account balance with a transaction. Such services are typically *idempotent*: i.e., they will not change their state should a subsequent interaction be attempted for the same transaction. The behavioral model of the account update service then consists of an initial communication – incorporating the transaction to record – followed by a response which might include the new balance.

2.3.2.1 Process Model

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.

For example, in a publish-and-subscribe service, there are individual operations dealing with registering a new subscription and publishing a new notice. A longer-term view of this service would consider the total sequence of notifications associated with a given subscription. Another concept that may be featured in a process model is the transactional structure of a service (c.f. ACID analysis of processes).

Note that although the existence of a process model is necessary to this Reference Model, its extent is not defined. In some architectures the process model will include aspects that are not strictly part of this reference model – for example 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.

Comment [m12]: Used to be "fundamental". Fundamental was a bit strong, IMO.

Choosing an appropriate representation of process models is a complicated business, made more so because processing such representations quickly becomes intractable for non-trivial process models. For example, the task of comparing two processes is a difficult exercise that is provably impossible in the general case. On the other hand, without some such expressive power it can be difficult to capture the required dependencies that are a natural part of process descriptions.

However, showing that two process models are equivalent is not the only requirement for representing process models. A more common requirement is simply to be able to identify the appropriate steps that must be followed for a successful interaction. This is analogous to following a recipe or executing a program – a task that is easily mechanizable.

2.3.2.2 Behavior

The **behavioral** model of a service is about the behavior that results in interactions with the service. Of course, a great portion of the behavior of a service may be private; however, the expected public view of a service surely includes the implied behavior of the service.

Comment [AE13]: Does it? It may imply it only if one thinks it implies such. In itself, I think a service's behavior is somewhat inert.

Comment [FGM]: Yes, informally. E.g., one might imagine that updating a bank account with a transaction is part of the expectations associated with the bank service; even though the outsider can never see that update going on.

For example, in a service that manages a bank account's state, it is not sufficient to know that to use the service you need to exchange a given message (with appropriate authentication tokens). It is also of the essence that using the service may actually affect the bank account – withdrawing cash from it for example.

The behavior of a service is closely connected to its intended real-world effect; although not identical to it. In general, we can state that the behavior of a service (an attempt to withdraw cash from an account) results in an intended (or occasionally unintended) effect in the world: the account's balance is lower.

2.3.3 Services in context

In an implementation, services are associated with an **execution context**. Consider, for example, that there is a distinction between a potential service and an actual service that is capable of being interacted with. An actualized service has an execution context that determines many of the properties of the service; including attributes such as security.

Comment [FGM: I would prefer something like Running services, or executing services.

Suppose that it were important that a given service was always executed in an authenticated context – i.e., that the service provider and the service consumer have authenticated themselves to each other. The details of how authentication is performed are not our concern here. That authentication context is an example of a particular execution context that applies to the service.

Comment [m16]: Peter suggests: "The details of how authentication is performed are the concern of the reference model. However, such an authentication requirement is an example of a particular execution context that applies to given service and its implementation."

I don't think I agree. Rejected unless someone else speaks up.

The execution context is a touchstone for many aspects of the service – what policies are in force for example, whether it is available, and so on.

2.4 Policies and Expectations

In the absence of intimate knowledge of the implementation of service providers and consumers a way of characterizing the use of a service is via the concepts of **policies** and **expectations**. We can understand what it means to interact with a service by examining the conditions on the use of the service and the expected results of using it.

Broadly speaking, a policy represents some form of constraint or condition on the use, deployment or description of an owned entity. We are focused primarily on the concept of policy as it applies to services.

On the other hand, the expectations associated with a service revolve around the consequences of interacting with the service. Normally, there is an expected **real world effect** as a result of

using a service – such as depositing money into an account. However, it can be difficult to characterize the real world effect of using a service, since the actions performed by a service implementation are inherently private to that provider.[1] Instead it may be more effective to consider the expectations for future interactions with services.

2.4.1 Service Policy

Abstractly, 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 **information** and **process 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 constraining the forms of messages. As an assertion, it is measurable: it may be true or false depending on whether the traffic is actually encrypted or not. Note that 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.

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

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. From a conceptual point of view, service policy enforcement amounts to ensuring that the assertion is consistent with the real world. An unenforceable policy is not really 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 may be, but need not be, written down in a formal machine processable form. The importance of such a machine processable form of policy depends on the purpose and applicability of the policy. In particular, where a policy declaration might affect whether a particular service is used or not, then such policies should be expressed in machine-processable form.

Languages that permit policy assertions also range in expressivity from simple propositional assertions to modal logic rules. However, the Reference Model is neutral to how a policy is represented.

A natural point of contact between service participants and policies associated with the service is in the service description – see Section 2.2.2. It would be natural for the service description to contain references to the policies associated with the service.

2.4.2 Services and expectations

There is nearly always a particular purpose associated with interacting with a service – the service consumer is trying to achieve some result by invoking 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 seats on a flight.

However, inherent to the concept of SOA is an arm's length approach to the relationship between service providers and consumers where there are minimum assumptions made by consumers about how a service is provided, and conversely minimum assumptions made by service providers about the connectivity of consumers. This separation is key to achieving large-scale systems and also to managing the evolution of such systems.

In keeping with this assumption, a more effective way of capturing the purpose of using a service is via the concept of **expectations**. I.e., rather than trying to ensure that the airline reservation service has recorded our booking, we are more interested in knowing that when we arrive at the airport, the airline will agree that we do indeed have a seat on the flight.

Expectations revolve around communication and future interactions much more than present time actions. Of course, in order for the airline to know that the seat is confirmed it will likely use some kind of system for recording the reservation; but, by minimizing assumptions about how the airline fulfils its contracts, we maximize the potential for smooth interoperation.

It is natural to characterize the expectations associated with a service interaction in terms of the message traffic exchanged with the service. In a manner that is completely analogous to the service interface, we can define expectations in terms of the kinds of information that will be provided subsequently by a service – as opposed to the information that is required for a current interaction. For example, a successful interaction with a courier or package delivery service might result in a tracking number. The expectation is that presenting that tracking number to the appropriate service will result in information about the current whereabouts of the package being delivered.

The expectations arising from a use of a service may be described in much the same ways that policies are described, except that the natural container for this is the **service contract.**

2.4.2.1 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, but need not be, expressed in a machine processable form. 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 – that would facilitate 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.

Comment [FGM: Have deleted text on contract life cycle – as not being critical to the SOA itself.

A variant of the policy concept is the agreement or **contract**. A contract has all the same features as a policy with one key addition: the concept of agreement – contracts are policies that have been agreed to by participants governed by the policy; policies do not need agreement only enforcement.

2.5 Service discoverability

A key concept of the SOA Reference Model is the discoverability of services; which is an important aspect of bringing together the service provider and consumer: A service provider must be capable of making details of the service (notably service description and policies) available to potential customers; and customers must be capable of finding that information.

Comment [FGM: Again, it may **not** be the service provider that provides this description. Ask Wal-Mart or GM

This might (and commonly does) involve a service provider entering the service description into a service registry and the service consumer searching for an appropriate match to their needs. The SOA concept of discoverability is not restricted to any single mechanism. In a pure P2P architecture there would be no registry at all.

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.

Comment [FGM: May also not be automated. The URL for the service may be *well-known* – to be found on the side of a bus.

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 as a prelude to evaluating whether the service matches their expectations and, if so, how to proceed to establish a contract and invoke the service.

In some contexts there are advantages to a registry model in which service capabilities are advertised, and then matched against requests of service. Such models should ideally allow for the capture of information both of service offers and service requests, so that it can be used in a reverse registry that records needs and queries on offers as well as, or instead of, records of service offers.

Comment [FGM: There are also significant scalability issues with registries.

Specific SOA reference architectures and implementations will prescribe the mechanisms for actual service discovery, ensuring a service’s presence and availability, and failure conditions and error handling.

Comment [FGM: I have redrafted this to better reflect the nature of a reference model.

3 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 practice* 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.

4 References

4.1 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.

4.2 Non-Normative

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

Appendix A. Glossary

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.

Advertising (or Announcement of Availability)

A means of conveying the existence of and sharing awareness about a service to potential consumers.

Agent (requester or provider)

An entity acting on behalf and with the authority of another entity and charged to fulfill a task.

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 an agent establishes – to an agreed level of confidence – the identity of another entity.

(Service) Consumer

An entity which intends to make use of a service.

Contract

The syntactic, semantic and logical constraints governing the use of a service.

Data Model

A Data Model is the abstract paradigm used in the invocation and consumption of a service. It is expressed as a set of information items associated with the use of a service.

Discovery

The act of detecting and gaining understanding of the nature of a service.

Encapsulation

The act of hiding internal specifications of an entity from the user of that entity, in such a way that the internal data and methods of the entity can be changed without changing the manner in which the entity is used. What is seen by the user is only an interface, or service.

Framework

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

Interface

A named set of operations that characterize the behavior of an entity.

Mediation

The transformation, routing, validation and processing of messages.

Message

A serialized set of data that is used to convey a request or response from one party to another.

Metadata

754 A set of properties of a given entity which are intended to describe and/or indicate the nature and
755 purpose of the entity and/or its relationship with others.
756
757 Negotiation
758 A process that seeks to establish an acceptable basis for a contract between agents for the
759 provision of a service.
760
761 Ontology
762 Represents an agreement within a specific environment of the meanings to be associated with
763 different concepts and their relations to each other.
764
765 Opaqueness
766 The extent to which an agent is able to interact successfully with a service without detecting how
767 the service is implemented.
768
769 Policy
770 A statement of obligations, constraints or other conditions of use of a given service. When a
771 specific set of entities accept such a policy, a contract is usually established.
772
773 Reference Model
774 A reference model is an abstract framework for understanding significant relationships among the
775 entities of some environment that enables the development of specific architectures using
776 consistent standards or specifications supporting that environment.
777 A reference model is based on a small number of unifying concepts. A reference model is not
778 directly tied to any standards, technologies or other concrete implementation details, but it does
779 seek to provide a common semantics that can be used unambiguously across and between
780 different implementations.
781
782 (Service) Requester or provider
783 An agent that interacts with a service in order to achieve a goal
784

Security

A set of policies and measures designed to ensure that agents in an environment can only perform actions that have been allowed. Security in a specific environment is an agreed compromise between meeting the needs of agents and maintaining the integrity of the environment.

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

A behavior or set of behaviors offered by one entity for use by another according to a policy and in line with a service description.

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;

- the procedures and constraints governing invocation of the service,

and thus determine whether the service meets the expectations and requirements of the consumer. Acceptance of the service description by a consumer does not of itself imply a contract to use the service.

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. This enables rapid and economical implementation in response to new requirements thus ensuring that services respond to perceived user needs.

SOA uses the object-oriented principle of encapsulation in which entities are accessible only through interfaces and where those entities are connected by well-defined interface agreements or contracts.

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.