Service Oriented Architecture Reference Model

Working Draft 05, 03 May 2005

Document identifier:

wd-soa-rm-05

Location:

http://www.oasis-open.org/

Editors:

C. Matthew MacKenzie, Adobe Systems Incorporated, mattm@adobe.com
Christopher Bashioum, Mitre Corporation, cbashioum@mitre.org
Ken Laskey, Mitre Corporation, klaskey@mitre.org
Wesley McGregor, GoC(ed: ?), McGregor.Wesley@tbs-sct.gc.ca
Francis McCabe, Fujitsu (Soft? Consult?), fgm@fla.fujitsu.com
Don Flinn, Individual, flinn@alum.mit.edu
Peter Brown, Individual, peter@justbrown.net
Vikas Deolaliker

Abstract:

This Service Oriented Architecture Reference Model is an abstract framework for understanding significant entities and relationships amongst 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 services oriented architectures or for education and explaining 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 system 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

1 Introduction

The service-oriented architecture (SOA) paradigm has received significant attention within the software design and development industry in recent times resulting in many conflicting definitions of service-oriented architecture. 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.

This document explicitly avoids defining implementation detail, as doing so would unnecessarily constrain and date the reference model. The goal is to provide a document that can stay relevant through the various technology evolutions that we experience in this industry.

A reference model cannot be implemented, nor should it be. A reference model is a foundational work that can and should be used to develop architectural patterns and promote effective discourse on derived works.

1.1 Audience

The intended audiences of this document non-exhaustively include:

- 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.

1.2 How to Use the Reference Model

New readers are encouraged to read this reference model in its' entirety, from beginning to end. Concepts are presented in an order that the authors hope promote understanding, quickly.

Section *0* introduces the conventions 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 service oriented reference model. A brief overview of the components and their relationships is given. The following subsections delve into greater detail on each component, including their externally visible properties and relationships to each other. 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 aims to provide guidelines for conformance with the reference model and is aimed at those who wish to explicitly state that their architectures are conformant with this reference model.

The appendixes provide several non-normative examples and a glossary to provide clarity of terms whose use may otherwise be ambiguous.

1.3 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]**.

1.4 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 this SOA Reference Model of this TC as a base or inspiration for their work when complete.

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.

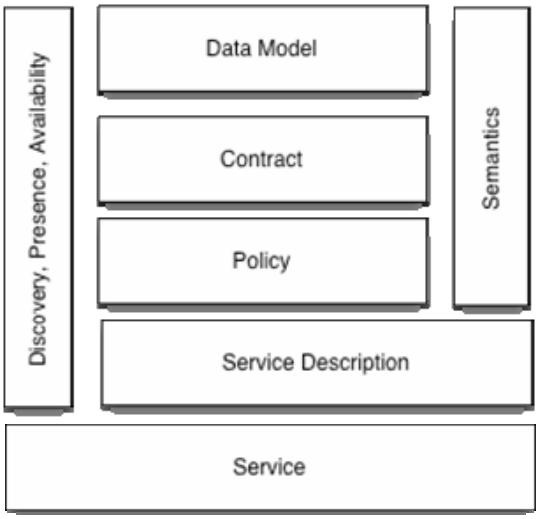
2 The Reference Model

[ed: This text was provided by Duane, and does require some stylistic rework.]

Figure 1 - SOA Architectural Model introduces the core service oriented architecture reference model and its high level components. Services are the fundamental base component of service-oriented architectures. Each Service has a Service Description. A Service Description is a set of metadata declaring all aspects of a service necessary for a Service Consumer to understand the service's externally inspect-able aspects. A Policy is a set of assertions that must be adhered to when a service is invoked. A Contract is implied when a Service Consumer makes and invocation request to a service, in substantial alignment with the Policy declaration.

A Data Model is the abstract paradigm used in the invocation and consumption of a Service. A Data Model will likely manifest itself within a concrete architecture as a set of concrete Messages.

Semantic agreement on what entities mean with respect to their roles in a system is necessary for service-oriented architecture. Many of the components (Service Descriptions, Policies, Contracts and Data Models) need to be available for discovery by potential service consumers to determine both the suitability of a service and their ability to invoke and/or consume the service. The concept of Discovery is to gain awareness of the Presence of the elements and details of their availability.



159

160 *Figure 1 - SOA Architectural Model*

161

162 2.1 Services

163

164 A Service is a set of functionality provided by one entity for the use of others.

165

166 There is no need to make architectural distinctions between services that are consumed as part
167 of a process vs. ones that are not.

168

169 There is not a one to one correlation between requests to invoke a service and instances of a
170 service being consumed.

171

172 Opacity is a core component of services.

173

2.1.1 Service Composition

Since services are opaque, a Service Consumer cannot see anything beyond it. If one service is actually consuming and aggregating two other services, the Service Consumer cannot and should not know such. Whether a Service's functions are mapped to a set of classes in some native language or another service is not important or relevant (other than the service metadata stating what invoking the service means or does)

Figure 2 - Service Composition

Examining Figure 2 - Service Composition above, the service function (for service A) is described in the service description specific to that service. If completing the function depends on two or more serial or parallel paths of execution successfully completing behind the service interface (like calling services B and C) within a certain time frame, that is not relevant to state in the service description for service A. The service consumer is only concerned with the service's ultimate success or failure. Mapping the functionality to success and failure is the responsibility of the service provider. This is necessary to preserve the axiom of opaqueness.

The functionality described above is mandatory to comply with the notion of service autonomy. A service alone must determine whether an invocation request succeeds or fails.

Note (non-normative) If a service consumer can see any specifics behind the service, this violates several of the core principles of SOA. If visibility beyond the offered service is required, then the service does not meet the demand of the service consumer. Accordingly, the service provider and consumer should discuss and re engineer the service.

Note

When implementing, more complex patterns of service invocation can be facilitated while keeping these three axioms. If a transaction sequence is needed, a service interface can offer two services - a put() and a commit().

2.1.2 Service Description

Each logical Service has exactly one canonical Service Description.

A Service Description is comprised of three parts:

a. Data Model - The logical expression of a set of information items associated with the consumption of a service or services;
b. Policy - Assertions and obligations that service consumers and/or providers must adhere to or provide; and
c. Contract (and/or offer thereof) - the syntactic, semantic and logical constraints governing on the use of a service.

2.2 Policies and contracts

Broadly speaking, a policy represents some form of constraint or condition on the use, deployment or description of an owned entity. Policies are inherently unilateral – any participant may have policies about issues that are important to them. A contract, however, is a policy that has been agreed to.

Where a contract can refer to everything from the detailed description of the service interface to the legal contract entered into when two or more parties use a service. However, the SOA RM focuses on those agreements necessary for a successful interaction with a service.

2.2.1 Service Policy

Abstractly, a policy is an assertion that expresses intent on the part of a participant.

Policies apply to many aspects of SOAs: to security, to privacy, manageability, Quality of Service and so on.

Policy assertions may be, but need not be, written down in a formal machine processable form. Languages that permit policy assertions also range in expressivity from simple propositional assertions to modal logic rules. However, the SOA RM 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. It would be natural for the service description to contain references to the policies associated with the service.

Associated with policies is the concept of enforcement. Enforcement is the realization of the policy: an un-enforced policy is simply an abstract logical proposition. However, how a policy is enforced, or even whether a policy is enforced is not a relevant part of the reference model.

A policy always represents a participant's point of view. For example, a provider of a service may have a policy that all users of the service must be authenticated prior to their access to certain functions. This policy is one that may be enforced by the service provider independently of any agreement from potential users of the service. Similarly, someone's agent may embody a privacy policy independently of any services the agent interacts with.

2.2.2 Service Contract

Where a policy represents an assertion from the point of view of a participant, 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. However, the concept of a service contract within the SOA RM applies primarily to the requirements for the successful use and provision of services.

A contract may be, but need not be, expressed in a machine process-able form. It seems significantly likely that an executed contract will not be in a machine process-able form; especially for commercial agreements. However, languages that can express policies, especially the more powerful variants can often also be used to express machine process-able contracts.

Each contract may be associated with a life-cycle. This life-cycle has three main phases: a negotiation phase, an active phase and a completion phase.

While it is possible that a specific negotiation phase precedes an agreement to a contract, often it is more implicit. For example, merely attempting to interact with a service may represent an agreement to follow the prescribed procedures for using the service.

Often a contract specifies policies that are assumed to be in force during the active phase of the contract. As such, those policies are subject to enforcement in a similar that unilateral policies are.

Enforcement of an agreement will depend on the nature of the agreement: violating an infrastructure-level agreement is likely to lead to to errors and unexpected results. Violating a commercial agreement is likely to lead to loss of service or other legal remedies.

While there may be many kinds of contract, we envisage three main kinds of contract that may apply in service oriented architectures: the contracts that represent the valid use and provision of services, the contracts that represent the permitted uses of services and the contracts that result from using services.

For example, the service description may contain descriptions of the interfaces of a service – the kinds of data entities expected and the names of the operations supported – and may also contain choreographic descriptions of the order of interactions. Such descriptions may range from simple identifiers implying a mutually understood protocol to a complete description of the vocabularies, expected behaviors and so on.

However, a valid use of a service is not equivalent to a permitted use of the service. For example, one may present a syntactically correct request to a service for withdrawing money from an account. If that request is not accompanied by a suitable authentication, then that request is typically denied – it is not permitted. Many security considerations and quality of service considerations lie in this realm of agreement.

Often the purpose of interacting with a service is to effect a further agreement. For example, one use of a book-selling service is to cause a book to be purchased and delivered.

This kind of contract is an important aspect of the rationale for deploying Service

Oriented Architectures; however, such contracts are beyond the scope of this SOA RM.

2.3 Semantics

The semantics of a service are the shared expectations about the service. Fundamentally, we expect that all services deployed in a SOA have an intended purpose. That purpose is the linchpin by which we measure the expectations for a service and is the basis of its semantics. The purpose of a service is the highest level semantic characterization of the service.

In principle, the semantics of a service many aspects of its establishment – from the format and structure of any data communicated between the participants of a service interaction to the stateful requirements on the participants to the expected effects of successfully interacting with the service.

One of the hallmarks of a Service Oriented Architecture is the degree of documentation associated with it. The purpose of this metadata is to facilitate integration, particularly across ownership domains. By providing descriptions, the task of designing client applications that make use of a service is considerably enhanced.

In this spirit, we might also expect that the semantic aspects of a service may also be documented. Such documentation will, in principle, be layered into several levels:

- The metadata required to reliably contact the service and to establish communication with it. In Web Services, this role may be filled by descriptions using the WS-Reliability specification.
- The metadata required to reliably format data for interchange between service participants.
- In Web Services, this role may be filled by WSDL documents.
- The metadata required to reliably sequence operations of the service. Documents using specifications such as WSBEPL and CDL are oriented towards such requirements.
- The metadata required to adequately measure the effect of using a service and of the requirements of the participants. Often, this is the kind of description labeled as semantic, although, in reality, all the above documents represent descriptions of the semantics of the service – albeit at different levels of abstraction.

- There may also be documents that relate to any policies governing the service and to any agreements and contracts associated with the service. Such documents may range in scope from simple technical policies to legal contracts valid in international law.

If documented in metadata, a service's semantics has many possible uses: it 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.

However, it is not essential to the concept of SOAs that the semantics of a service be so completely described.

2.3.1 Data/Information Model

[vikas deolaliker]

2.4 Discovery, Presence and Availability

[ed: title was changed, text below needs to jive.]

The main concept is a methodology or mechanism to convey awareness of (the existence of) a service(s) to all consumers on a fabric.

Advertising makes discovery possible.

A Service Description is advertised to consumers on a fabric to make it discoverable.

Discovery does not constitute authorization to execute against the service.

[from W3C WSA] Discovery is the act of locating a resource description

Discovery involves matching a set of functional and other criteria with a set of resource descriptions.

Discovery may be performed by an agent, or by an end-user

Discovery may be realized using a discovery service [end W3C WSA]

3 Conformance Guidelines

The authors of this reference model envision that architects may wish to declare their architecture is conformant with this reference. In order to be conformant to this reference model, a mapping must be made from each core element of this reference model to components of the conformant architecture.

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.

Appendix A. Glossary

Several terms are used within this Reference Model are also used in other specifications. This glossary locally scopes the semantics of those terms where ambiguity exists or overrides those definitions.

Advertising

A methodology to convey awareness of (the existence of) a service(s) to all consumers on a fabric. Advertising makes discovery possible.

Agent (requester or provider)

An entity acting on behalf of another entity to fulfill a task.

Architecture

Software architecture for a system is the structure or structures of the system, which consist of elements and their externally visible properties, and the relationships among them.

Service Consumer

An entity which makes use of a service.

Contract

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

Data Model

The logical expression of a set of information items associated with the consumption of a service.

Discovery

The act of gaining knowledge of a logical service, its existence and details of how to use it.

Interface

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

Message

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

Policy

Policy is a statement of obligations, constraints or other conditions of service use.

A contract is formed when a specific set of entities accept a policy.

Requester or provider

Person or organization involved in an SOA transaction an agent that interacts with a service in order to achieve a goal

Security

Computer security is the effort to create a secure computing platform, designed so that agents (users or programs) can only perform actions that have been allowed. This involves specifying and implementing a security policy. The actions in question can be reduced to operations of access, modification and deletion. Computer security can be seen as a subfield of security engineering, which looks at broader security issues in addition to computer security. (from Wikipedia)

Semantics

Shared conceptualization of the implied meaning of information. Represents a contract governing the meaning and purpose.

Service

A behavior, or set of behaviors provided for use by another entity.

Service description

A specification of the information necessary to a) allow a potential consumer to determine whether or not this service is applicable, and b) facilitate invocation.

Service Oriented Architecture (SOA)

A form of Enterprise Architecture. The difference between Enterprise Architecture and SOA lies mostly in the fact that EA is specific to an enterprise, while SOA can be abstracted out of a given Enterprise, and collected along with other SOA components so abstracted to form a registry of available services SOA is potentially a specialization of a combination of many things - interface based design (IBD), component architecture (CA), OO methodology etc.

Service Oriented Architecture Reference Model (SOA-RM)

A reference model is an abstract framework for understanding significant relationships among the entities of some environment, and for the development of 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. Is not architecture for a single implementation. Is a model for developing a range of Service Oriented Architectures and analysis/comparison thereof. Is a framework for understanding significant relationships among the entities in an SOA environment. DISCUSSION POINT: should the word “elements” be used in place of “entities” above? Is based on a small number of unifying concepts of all SOAs. A Reference Model is the best mechanism to define SOA.

Appendix B. Use Cases and Examples (Non-Normative)

Appendix C. Acknowledgments

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

[TODO: insert cte. Members]

Appendix D. 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.