Reference Ontology for Semantic Service Oriented Architectures Version 1.0

Public Review Draft 02

21 April 2011

Specification URIs:
This Version:
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/pr02/see-rosoa-v1.0-pr02.doc (Authoritative)
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/pr02/see-rosoa-v1.0-pr02.html
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/pr02/see-rosoa-v1.0-pr02.pdf

Previous Version:
N/A

Latest Version:
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/see-rosoa-v1.0.doc
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/see-rosoa-v1.0.html
http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/see-rosoa-v1.0.pdf

Technical Committee:
OASIS Semantic Execution Environment TC

Chair(s):
Barry Norton, Karlsruhe Institute of Technology <barry.norton@kit.edu>
John Domingue, Open University, <j.b.domingue@open.ac.uk>

Editor(s):
Barry Norton, KIT, <barry.norton@kit.edu>
Mick Kerrigan, STI, <mick.kerrigan@sti2.at>
Adrian Mocan, SAP, <adrian.mocan@sap.com>
Alessio Carenini, CEFRIEL, <alessio.carenini@cefriel.it>
Emilia Cimpian, STI, <emilia.cimpian@sti2.at>
Marc Haines, Individual <mhaines@uwm.edu>
James Scicluna, STI, <james.scicluna@sti2.at>
Michal Zaremba, STI, <michal.zaremba@sti2.at>

Related work:
OASIS Reference Model for Service Oriented Architecture V 1.0
Semantic-ex Background and Related Work

Declared XML Namespace(s):
http://docs.oasis-open.org/semantic-ex/ns/referenceontology/v1.0

Abstract:

This Reference Ontology for Semantic Service Oriented Architectures is an abstract framework for understanding significant entities and relationships between them within a Semantically-enabled Service-Oriented environment. It may be leveraged for the development of related standards or specifications supporting that environment, as well as guiding efforts to realize concrete solutions.

This Reference Ontology builds on the OASIS Reference Model for Service Oriented Architecture (SOA-RM) and combines it with the key concepts of semantics that are relevant for Semantically-enabling Service Oriented Architectures.

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

Just as the SOA-RM, this reference ontology focuses on the field of software architecture. The concepts and relationships described may apply to other "service" environments; however, this specification makes no attempt to completely account for use outside of the software domain.

Status:

This document was last revised or approved by the Semantic Execution Environment Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/semantic-ex/.

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 Technical Committee web page (http://www.oasis-open.org/committees/semantic-ex/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/semantic-ex/.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

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 section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, 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 OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

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' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on 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 OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

Although Service Oriented Architectures (SOAs) have gathered a lot of attention within business organizations, for a long time there was no clear understanding of what an SOA precisely is. As a result reference models have been published to define SOA; we note particularly the OASIS SOA Reference Model [1]. However, with the emergence of **Semantic Web** technologies, in particular **Semantic Web Services (SWSs)**, new breeds of SOAs are being developed, namely **Semantic Service Oriented Architectures (SSOAs)**. Semantic Web Services provide a means for created better descriptions for Web services, with fully semantics. As such Semantic Web Services are a layer on top of existing Web service technologies and not a replacement for them. SSOAs use semantic technologies to advance solutions to problems by which traditional SOA, focusing on WSDL-described services, are limited. They provide a means for further automation for service consumers' tasks, particularly service discovery, selection, composition and execution, as well as easing general interoperability issues between services.

In order to use the semantic descriptions present in a SSOA to automate such SOA features, a set of platform services that provide this automation functionality are required within the SSOA. These services are collectively termed a **Semantic Execution Environment (SEE)** for Semantic Web Services, with a SEE being at the core of a SSOA. There are a number of different implementations of SEEs currently under development in the research community, which have some common features. Thus the purpose of this document is to define an extended reference model for SSOAs, as supported by SEEs. This model will be defined formally using an ontology. The aim of this ontology is to provide a point of reference formally specified so that it can support the definition and development of SSOAs.

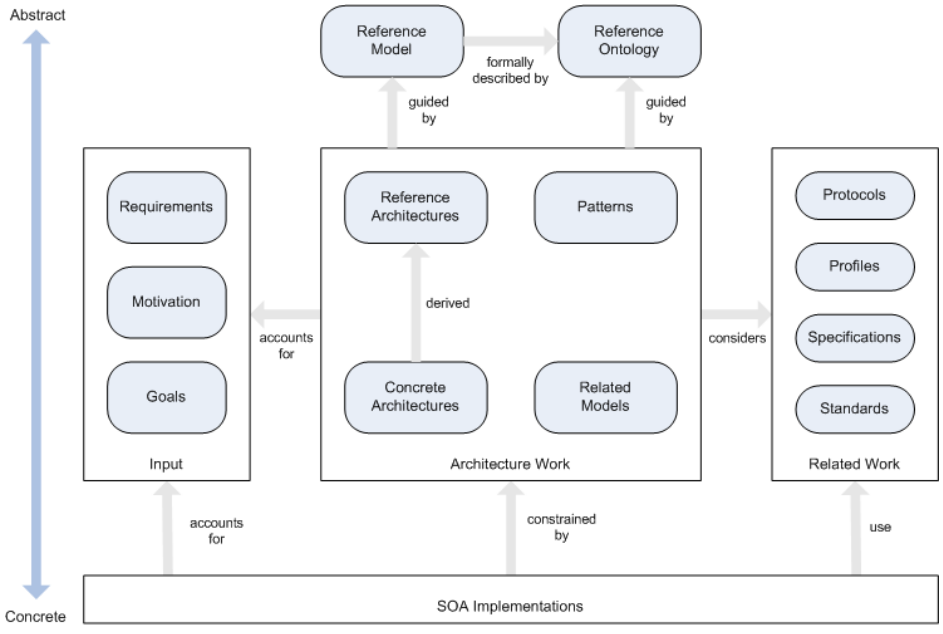


Figure 1-1 – Relationship of the Reference Ontology to Other SOA Specifications and Standards

Figure 1-1 depicts how the Reference Ontology relates to other pieces of work within the SOA community. The figure is derived from Figure 1 in the SOA Reference Model document [1], and introduces the Reference Ontology alongside the Reference Model element. The Reference Ontology presented in this document is a further step towards formalization of the Reference Model but also accommodates the extensions associated with Semantic Web Services resulting in Semantic SOAs. Since the start of this work, the SOA-RM committee has also started work on a Reference Architecture, which also aims at further formalisation of the reference model, but we consider ontologisation central to the semantics-based approach and diverge. Indeed when we say Reference Architecture we shall refer to a reference architecture for SEEs, not to the SOA Reference Architecture. Furthermore when we say

Concrete Architectures we refer to implementations of semantics-enabled SOAs such as WSMX [2], IRS III [3], and METEOR-S [4].

The Related Models in Figure 1 include, for us, the Web Service Modeling Ontology (WSMO) [5], Semantic Annotations for WSDL and XML Schema (SAWSDL) [8], the Web Ontology Language for Services (OWL-S)[1] [9], and the Semantic Web Services Ontology (SWSO) [10]. Patterns fulfill the same role in Semantic- as in pre-Semantic- SOA, which is to say that they define more specific categories of service-oriented designs. The Protocols and Profiles (those considered as part of the related work) are the same as for W3C WS-Stack based SOAs. However, with respect to Specifications and Standards, we further take into account emerging Semantic Web Languages such as the OWL, RDF and RIF standards from W3C, and the WSML and SWSL languages for describing services semantically. These languages play a very important role since they are the pillars of Semantic Technologies. The Input features (Requirements, Motivation and Goals) are the same as for SOAs, with the addition that we have more emphasis on automation, as stated earlier.

1.1 Motivation and Scope

With the term “Semantic” we mean the formal (and thus unambiguous) description of some particular object (more in section 2), which is subject to automated ontology-based reasoning. Within the context of the Reference Ontology, these objects are mainly the data handled by the services and the services themselves. Semantic descriptions within SOAs allow reasoning tools to automate tasks. More specifically, semantics help in the following ways:

- Formally and unambiguously define the data models and processes underlying the system;
- Allow automated discovery and composition of services;
- Automatically resolve data and process mismatches, easing integration and improving interoperability;
- Ease the process of service ranking, negotiation and contracting.

The scope of this document is therefore to provide an ontology that formally describes the different elements comprising a SSOA in order to achieve the above objectives.

1.2 Audience

The target audience for this document extends that of the SOA RM; however we provide an exhaustive list in order to keep the document self-contained:

- Architects and developers designing, identifying or developing a system based on Service Oriented Architectures;
- Standards architects and analysts developing specifications that rely on Service Oriented Architecture concepts;
- Decision makers seeking a "consistent and common" understanding of Service Oriented Architectures;
- Users who need a better understanding of the concepts and benefits of Service Oriented Architectures;

[1] It may be noted that no unified Semantic Execution Environments exist for OWL-S; a list of the major, but separate, OWL-S tools is available as http://www.daml.org/services/owl-s/tools.html, which includes the OWL-S VM

- Academics and researchers that are researching within the Semantic Web and Semantic Web Service communities;
- I.T. consultants that provide businesses with support on Semantic technologies and SOAs in general.

1.3 Guide to this Document

It is assumed that readers who are not familiar with SOA concepts and terminologies read first the SOA Reference Model [1] document since this document builds on top of its concepts. Furthermore, readers who are new to the concept of Semantic Technologies are encouraged to read this document in its entirety.

Section 1 introduces the Semantic SOA Reference Ontology and how it relates to other work (in particular the SOA RM). It defines the audience and also provides a description of the notational conventions used in this document. Both of these elements are important in order for the reader to understand the content of the rest of the document.

Section 2 provides an overview of Semantics and how they interrelate with SOAs. It starts by describing the deficiencies of the classical SOA and the problems in building them. It then continues with examples and situations of how Semantic Technologies can help to overcome these deficiencies. Section 2 strengthens the motivations and objectives already described in this section.

Section 3 describes the SOA Reference Model [1] and builds on top of this by introducing new key concepts required for SSOAs. It first describes what we understand by a service followed by the dynamics of a service – how the service is perceived by the real world. Other related concepts are also described (including, for example, the behavior of the Web service). Section 3 shows the differences between the classical SOA RM and the SSOA RM and provides the necessary building blocks for specifying the Reference Ontology.

Section 4 defines the Reference Ontology for SSOAs. The ontology is first described using Concept Maps and UML Diagrams (notation described in Section 1.4 below). It is then formally described using RDFS [7] in Appendix B as explained in Section 1.4.2.

The glossary provides definitions of terms that are relied upon within the document. Terms that are defined in the glossary are marked in **bold** at their first occurrence in the document.

Note that while the concepts and relationships described in this document may apply to other “service” environments, the definitions and descriptions contained herein focus on the field of software architectures and make no attempt to completely account for their use outside of the software domain. Examples included in this document, which are taken from a variety of domains, are used strictly for illustrative purposes.

1.4 Notational Conventions

Throughout this document we use both Concept Map and UML Class Diagram notations to illustrate models, this is due to the derivation from – and preservation of links to – the SOA RM specification, which uses the former, together with the need to provide an accessible representation of the ontology-based model. For clarity these two notations are distinguished in the caption of the figures throughout the document; figures whose caption end with [Concept Map] conform to the Concept Map notation, while figures whose caption end with [UML] conform to the representation of ontologies in the UML Class Diagram notation, as described below. This document does not use the notation from RFC2119 0, for example MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL as cardinality constraints are present within the UML diagrams.

1.4.1 Concept Maps

The Concept Map notation used in this document is the same as for that in the SOA RM; however we give a brief description here to keep the document self-contained.

There is no normative convention for interpreting Concept Maps and other than described in this section, no detailed information can be derived from the Concept Maps.

Figure 1-2 - A basic Concept Map [Concept Map]

As used in this document, a line between two concepts represents a relationship whereby the relationship is not labeled but rather is described in the text immediately preceding or following the figure. The arrow on a line indicates an asymmetrical relationship, where the concept to which the arrow points can be interpreted as depending in some way on the concept from which the line originates. The text accompanying each figure describes the nature of each relationship.

1.4.2 Ontologies

Within this document we use UML Class Diagrams to illustrate the Reference Ontology; the underlying formal definitions are made in RDF(S) – in particular using the Turtle serialisation. This is for two reasons: first, we must use a language with well-founded semantics, capable of machine reasoning – the general motivation of work in the Semantic Web that has produced several ontology languages.

This document sticks as far as possible to the Description Logic-compatible features of RDFS, though it makes limited use of meta-classing, hence the Reference Ontology is largely compatible with OWL-DL reasoning. The Reference Architecture will attach Reference Ontology concepts to *goal* descriptions to allow the characterization of the components of a Semantic Execution Environment (the core services of a SSOA). For this reason the Web Service Modeling Language may be used. The Execution Scenarios will attach Reference Ontology concepts, and Reference Architecture goals, to descriptions of (IT) services to illustrate how the SEE components can work together to achieve common tasks. Finally, concrete architectures may be defined by linking concrete services to the goals from the Reference Architecture.

In the remainder of this section we sketch the relationship between UML Class Diagrams, as used within the text, to RDFS descriptions. In the following section we reproduce these definitions.

Classes

The fundamental feature of Class Diagrams – and indeed Object-oriented design (OOD), which is the real target of UML – are classes, which are shown as square boxes with their identifier listed inside. We use UML classes to represent RDFS classes. Where the namespace into which classes are defined is clear, we allow ourselves to omit this information in the Class Diagram. Where different namespaces are used, we use the notation for packages to make the namespace clear.

Figure 1-3 hence corresponds with Listing 1.

```
:A a rdfs:Class.

<http://www.example.com/ontologies/ns1#B> a rdfs:Class.
```

Listing 1: Example Concepts in Turtle

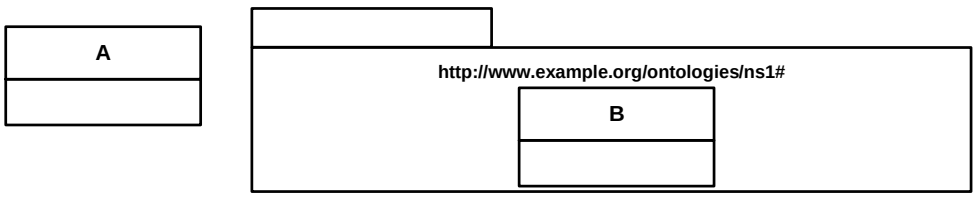


Figure 1-3: Representation of RDF Example Concepts in UML Class Diagram [UML]

While UML Class Diagrams allow the definition of operations and attributes within classes, we choose not to use these and always show classes with an undivided box.

Subsumption

The fundamental relationship between concepts in RDF(S), as with many ontology languages, is *subsumption*. This is represented by inheritance in UML Class Diagrams. Since we declare no operations there are thus no unwanted side-effects due to UML/OOD semantics; in particular there are no complications in the use of multiple parents for a given concept.

Figure 1-4 hence corresponds with Listing 1.

```
:B rdfs:subClassOf :A.

:D rdfs:subClassOf :A, :C.
```

Listing 2: Example of Subsumption between Concepts in RDFS

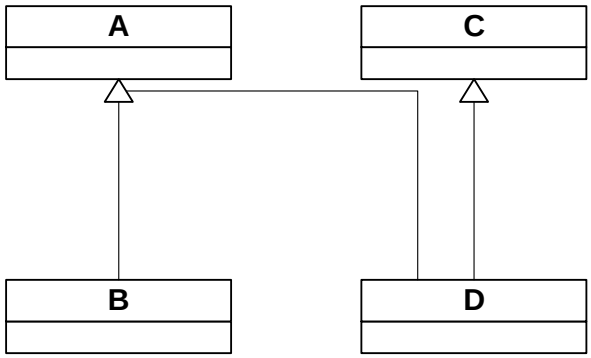


Figure 1-4: Representation of Subsumption Example in UML Class Diagram [UML]

Properties

The other explicit relationship between classes in RDFS is via *properties*. These are represented by (directed) *associations* in UML Class Diagrams, which is to say associations with a one-way navigability, so that the innavigable side of the association (or, more correctly, the end of unspecified navigability) is the class which is the domain of the property.

1.5 Terminology

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]**.

1.6 Normative References

[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.

1.7 Non-Normative References

[1] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz (eds.): Reference Model for Service Oriented Architecture 1.0, OASIS Standard, 12 October 2006, available at: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[2] A. Haller, E. Cimpian, A. Mocan, E. Oren, C. Bussler: WSMX: A Semantic Service-Oriented Architecture. In Proceedings of the International Conference on Web Services (ICWS 2005), Orlando, Florida

[3] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci. IRS-III: A broker-based Approach to Semantic Web Services, Journal of Web Semantics, 6, 2, pp. 109-132, Elsevier, 2008.

[4] World Wide Web, 6(2):109–132, 2008.K. Verma, K. Gomadam, A.P. Sheth, J.A. Miller, Z. Wu: The METEOR-S Approach for Configuring and Executing dynamic Web Processes. LSDIS Technical Report, 24 June, 2005, available at: http://lsdis.cs.uga.edu/projects/meteor-s/techRep6-24-05.pdf

[5] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, M. Stollberg: The Web Service Modeling Ontology WSMO.. Forschungsinstitut at the University of Innsbruck Technical Report, 16 February 2007, available at: http://www.wsmo.org/TR/d2/v1.4/

[6] S. Bradner, RFC2119 – Keywords for use in RFCs to indicate Requirement Levels, http://www.rfc.net/rfc2119.html

[7] H. Lausen and J. de Bruijn, A. Polleres and D. Fensel, WSML – A Language Framework for Semantic Web Services, Proceedings of the W3C workshop on Rule Languages for Interoperability, April 2005.

[8] J. Farrell, H. Lausen: Semantic Annotations for WSDL and XML Schema. W3C Recommendation, 28 August 2007, available at: http://www.w3.org/TR/sawsdl/

[9] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirinin, N. Srinivasan, K. Sycara: OWL-S: Semantic Markup for Web Services. DARPA DAML Program Technical Report, available at: http://www.ai.sri.com/daml/services/owl-s/1.2/overview/

[10] S. Battle, A. Bernstein, H. Boley, B. Grosof, G. Kruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, S. Tabet: Semantic Web Services Ontology (SWSO). DARPA DAML Program Technical Report, 9 May 2005, available at: http://www.daml.org/services/swsf/1.0/swso/

[11] D. Fensel, M. Kerrigan, M. Zaremba (eds.): Implementing Semantic Web Services - The SESA Framework, (Springer), 2008

2 Semantics and SOA

As noted in the Reference Model for Service Oriented Architecture (SOA-RM) standard, the notion of Service Oriented Architecture has received a lot of attention in the software design and development community. According to the SOA-RM, a "Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains." Service Oriented Architecture provides an architectural mechanism for addressing needs by composing unassociated capabilities that are accessible through SOA services. 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, by enhancing the ability of adapting applications more quickly to changes in market conditions and improving the reusability, modularity, composability and interoperability of functionality.

A service, in the context of SOA, refers to a mechanism that provides access to a capability that may have a real world effect or results in the exchange of information. Such services can be implemented leveraging many different standards and technologies, including but not limited to Web services using WSDL descriptions and SOAP messaging.

Building SOA-based systems using existing services still involves substantial human effort in the process of finding and using appropriate services. The need for human intervention can be attributed partly to the fact that standards that are typically used for describing services (e.g., WSDL), only focus on the syntactic aspect of the service interface, and provide little support for finding and using services that provide the appropriate desired functionality. In this "classical Web service" scenario, developers building an application, typically look for services that are available, either within their company's repository of services or in remote locations. Each time a need to invoke a service is identified, a set of candidate services must be found browsing in repositories (e.g. UDDI or ebXML repositories). While keywords and text search features can be leveraged to identify candidate service, the syntactically focused descriptions typically require evaluation by a human before a service can be used. In many instances further human interaction between the developer on the consumer side and the service provider is required to clarify the functionality and the meaning of the information that is being exchanged. Then tests can be performed on the candidate services. Finally, a service may be selected and integrated into the SOA-based system.

Not only is this process labor intensive, but the solution is fairly static, limiting the ability to adapt to changes quickly, which is a key promise of the SOA approach. Changes, whether it is new services that provide improved functionality or unavailability of currently used services, typically require human interaction in the classical integration scenario. The goal of a Semantically-enabled SOA is to add features that can help overcome these limitations and provide mechanisms to automate tasks that currently require human intervention.

2.1 Semantics

A key limitation of a "classical W3C WS-Stack based SOA", as mentioned above, is that the standards used for describing Web services provide very little detail about the service, beyond a simple description of the external interface they provide. With these descriptions it is impossible to provide further meaning about a service, such that reasonable inferences can be drawn regarding the functionality offered by the service, or the behavior of its outwardly facing interfaces.

Semantics is the study of meaning. A formal semantic description offers the opportunity of providing a mechanism for describing things more clearly and extensively. A formal semantic description is unambiguous within the context of the formalism and opens the opportunity for automated reasoning. Semantics come in many forms. Very basic advances towards semantics include annotations or tags that can be associated with an entity in order to give a description of what that thing is. Annotations or tags can be seen in action on sites like flickr.com®, where they are used for denoting what content appears in a particular picture or what a picture is about. This mechanism, of course, is very rudimentary and certainly not unambiguous in nature as annotations or tags are freeform in nature. To bring more meaning to the annotations, taxonomies can be introduced. Such structures give a mechanism for providing a controlled vocabulary of terms (i.e., a controlled set of annotations) and the relationship between them.

For example we can state that the concept of a *banana* is a sub class of the concept of a *fruit*. This additional semantic information enables us to reason about the semantic descriptions we have and make decisions based on the semantic descriptions, for example the query *"show me all photos containing a piece of fruit"* is posed, then those pictures that are annotated with the term banana would be found, as *banana* is a subclass of *fruit*. To add more semantics we can go even further and allow logical expressions to be added to taxonomies to turn them into ontologies, such that more complicated relationships between entities can be expressed. The addition of axiomatic information in this way also allows for much more sophisticated reasoning to take place and for new information to be inferred from existing information, for example the axiom *"all fruit is edible"* placed in a reasoner with the previous example would allow the fact *"bananas are edible"* to be inferred and thus queries like *"show me all photos containing things that are edible"* would find pictures of bananas.

2.2 Applying Semantics to SOA

As indicated earlier, the syntactically focused descriptions of services in a SOA-based system driven by the W3C WS-Stack limits the ability to automate tasks that are important for a quickly and reliably adapting to changes. The idea here is to apply semantics to the description of services within SOA and enhance service descriptions with additional semantic information that can be used in conjunction with semantic processing mechanisms (i.e., mediation).

By extending ontologies to describe services in a SOA-based system, a machine can reason about the functionality they provide, the mechanism to invoke them, and the data they expect as input and return as output. In other words each service that currently has a syntactic description (i.e., a WSDL document) will also have a semantic description in some formalism. Thus services within a Semantic SOA are not a reinvention of services, but an enhancement of them. In order to effectively describe services semantically we need to have an understanding of what elements need to be modeled within our semantic description. Within this document you will find the Reference Ontology for Service Oriented Architectures, which provides such a description of what elements need to be modeled in order to effectively provide semantic description for services and build a SOA-based system that is semantically-enabled, referred to as a Semantic SOA (SSOA).

Once services are described semantically, many of the tasks previously requiring human intervention in building and maintaining and application using SOA can be automated. For example, services can be *discovered* based upon the functionality they advertise in their semantic description, can be *selected* based upon the advertised (or observed) quality of the service, heterogeneity issues with respect to the data they exchange or the process to invoke them can be *mediated*. This allows for a SSOA, to dynamically bind to services at run time, removing the hard-wired behaviours that are typically for classical W3C WS-Stack based SOA systems. When new services appear on the market that fulfill functionality needed by the system, they can be considered alongside existing services that are being used already by the application and may be selected over these existing services based on the requirements of the application. Also if a given service that is usually used by the application is no longer available, it can be automatically replaced by another service that fulfills the same function.

3 Overview of SOA-RM

The notion of Service Oriented Architecture has been greatly used in the last couple of years by the software design and development communities. Yet, the various and very often conflicting definitions and terminology for SOA and its elements could hamper the adoption process and threaten the success and the impact of this technology. In order to provide a standard reference point in the design and implementation of SOA-based systems the OASIS SOA-RM Technical Committee[2] proposes an abstract framework for understanding the main entities and the relationships between them within a service oriented environment [1].

The resulting specification is a SOA Reference Model (SOA-RM), which is not directly dependent of any standards, technologies and implementation details. Its goal is to define the essence of Service Oriented Architecture, a normative vocabulary and a common understanding of SOA. The Reference Ontology takes this reference model as a starting point in defining the main aspects of a Semantically-enabled Service Oriented Architecture and it specifies how the normative elements of the SOA-RM can be augmented with semantics. As a consequence, this section gives a brief overview of the SOA-RM, along the several aspects it covers: the notion of *service*, the *dynamics of service* and the service-related concepts such as *service description*, *service execution context* and *service contracts and policies*, as shown in Figure 3-1.

3.1 What is a service?

SOA-RM defines a service as "*…a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.*" It identifies four main aspects regarding the service that have to be considered in any SOA:

- A service *enables access to one or more capabilities*;
- A service *enables access through at least one prescribed interface*;
- A service is *opaque to the service consumer* except for the information and behavioural models in the interface, and the information required to assess if a service meets the requesters needs;
- *Consequences of invoking a service* involving real world effect and possible change to the state of the service.

It is important to note that SOA-RM makes a clear distinction between the capability to which a service provides access (i.e., some functionality created to address a need) and the point of access where the capability can be consumed in the context of SOA.

3.2 Dynamics of Services

SOA-RM also provides guidelines regarding the interactions of the requester with a service. As such, among the service related concepts mentioned above, it identifies three fundamental concepts related with dynamics of the service: *Visibility, Interaction* and *Real World Effect* (see Figure 3-1).

[2] For more details, see http://www.oasis-open.org/committees/soa-rm.

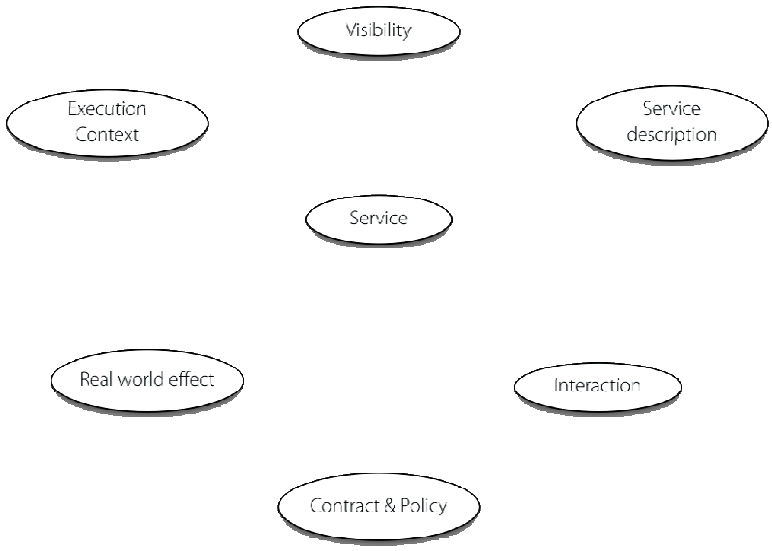


Figure 3-1. Fundamental Concepts of Service Dynamics (directly from [1]) [Concept Map]

Visibility in terms of SOA-RM is characterized in terms of *Awareness*, *Willingness* and *Reachability* (see Figure 3-2) where:

- *Awareness* is the state whereby the service requester is aware of the service provider or the other way around. It is normally achieved by having either the requester or the provider discovering the information the other party published in for example a public directory.
- *Willingness* concerns the intent to communicate. Even if the discovery process has been successful, without willingness to communicate from both requester and provider the interaction will fail.
- *Reachability* is the state that characterizes service participants that are able to interact, for example by exchanging information.

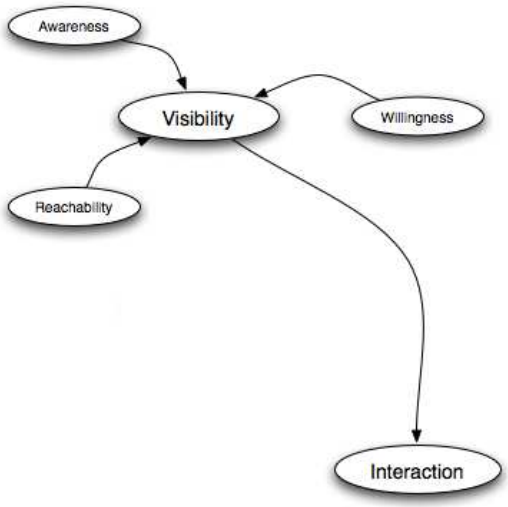


Figure 3-2. Service Visibility (adapted from [1]) [Concept Map]

The *interaction* with a service is reflected by the actions performed on the service, for example exchanging messages with the services. According to SOA-RM the key concepts affecting the interaction with a service are the following (see Figure 3-3):

- *Information Model* of a service characterizes the information that may be exchanged with the services and only descriptions of information that can be potentially exchanged with the service and their data structures are included in the information model. The information model can be also portioned in:
    - *Structure (Syntax)* refers to the representation, structure, and a form of information;
    - *Semantics* refers to the actual interpretation and intent of the data. Semantics becomes important especially when interaction occurs across ownership boundaries since the interpretation of data must be consistent between the participants in a service interaction.
- *Behavior Model* deals with *"knowledge of the actions invoked against the service and the process or temporal aspects of interacting with the service".* It consists of two distinct aspects:
    - *The action model* characterizes the actions that can be invoked against the service. Since a great part of the behavior implied by an action is private, the public view of the service includes the description of effects resulting from actions;
    - *The process model* defines temporal relationships of actions and events associated when interacting with a service. SOA-RM does not fully define the process model since it could include aspects that are not strictly part of SOA, e.g. orchestration of services.

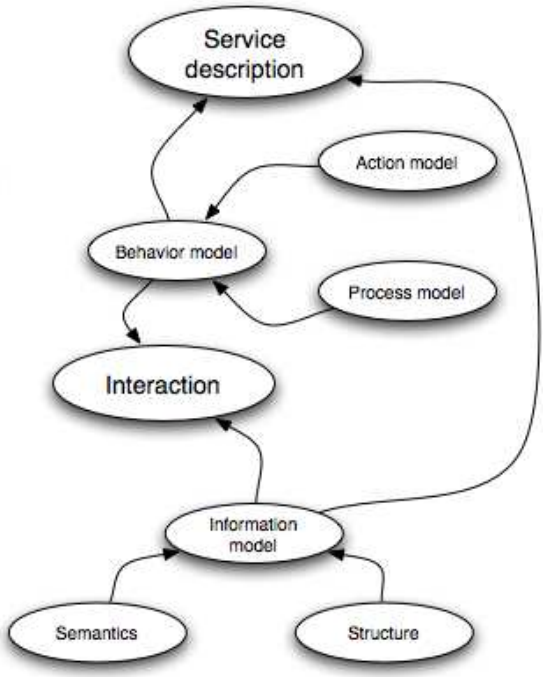


Figure 3-3. Service Interaction (adapted from [1]) [Concept Map]

The r*eal world effect* is the ultimate purpose associated with the interaction with a particular service. It can be the response to a request for information or the change in the state of some shared entities between the participants in the interaction.

3.3 Service Related Concepts

SOA-RM identifies a set of concepts crucial in enabling the interaction between a service consumer and a service. These concepts are the *service description*, the *service policies and contracts* and the *execution context.*

The *service description* encompasses the information needed in order to use the service (see Figure 3-4). The purpose of the service description is to facilitate visibility especially if the participants are part of

different ownership domains. By using the service description the service consumer should be able obtain the following items of information:

- Whether the service is reachable or not;
- Whether the service provides the function required by the requester;
- The set of policies the services operates under;
- That the service complies with the service consumer's policies;
- The means to interact with the service, including the format and content of the information to be exchanged, as well as the expected sequence of the information exchange.

As a consequence, there are several important aspects that have to be captured by the service description: the service reachability, the service functionality, the service-related policies, and the service interfaces.

- *Service reachability* is assured by including in the service description enough information to enable the service providers and services consumers to interact with each other. Such information could include service metadata (e.g. location, supported or required protocols), dynamic information about service (e.g. if the service is currently available), etc.
- *Service functionality* should be unambiguously captured by the service description and it should contain information about the function of a service and the real world effects that result from it being invoked. This piece of information should be expressed in a general-enough way to be understandable by service consumers while at the same time the vocabulary used should be expressive enough to capture the domain-specific details of the service functionality. Such information could include a textual description (for human consumption) or identifiers or keywords referencing machine-processable definitions.
- *Service-related policies* should be reflected by the service description in order to enable the prospective service consumer to determine if the service will act in a manner consistent with consumer's own constraints.
- The *service interface* describes the means to interact with the service. It could include specific protocols, commands and information exchange by which actions are initiated. It prescribes what information needs to be provided to the service in order to access its capabilities and interpret responses. This information is also referred as the information model of the service.

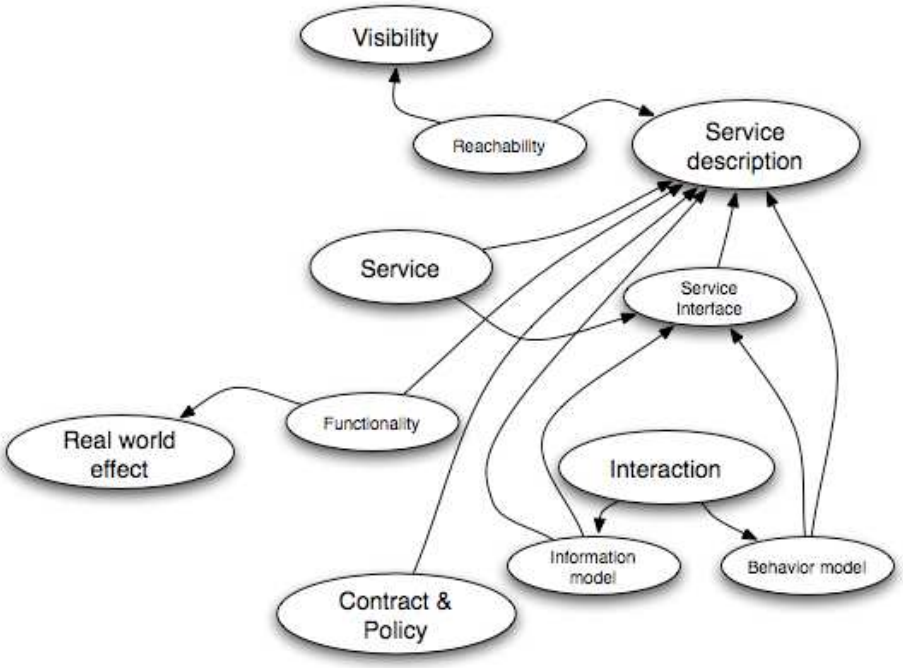


Figure 3-4. Service Description (directly from [1]) [Concept Map]

The *service policy* represents the constraints or the conditions on the use, deployment, or description of a service while a *contract* is an agreement on a measurable assertion that governs the requirements and expectations of two or more parties. Policies potentially apply to various aspects of SOA such as security, manageability, privacy, etc. but they could also be applied to business-oriented aspects, e.g. hours of business. In their turn contracts can as well cover a wide range of aspects of services: quality of services agreements, interface and choreography agreements, commercial agreements, etc. Note that the contract is derived from the service description; it can add consumer policies but cannot extend business or technical capabilities announced in the service description (though the capabilities may be detailed as needed).

The *execution context* represents the set of infrastructure elements, process entities, policy assertions, and agreements associated with a particular service interaction, forming a path between service consumers and service providers. The execution context is not limited to one side of the interaction but rather concerns the overall interaction, which includes the service provider, service consumer and the infrastructure in between.

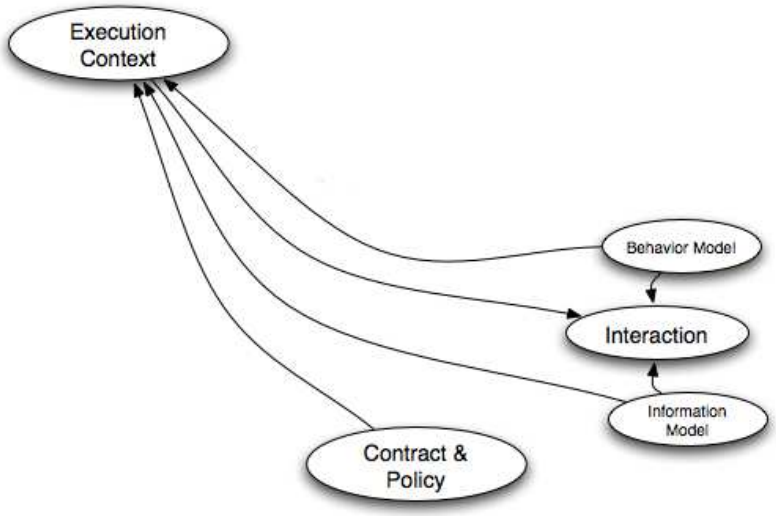


Figure 3-5. Execution Context (adapted from [1]) [Concept Map]

4 Reference Ontology for Semantic Service Oriented Architectures

The reference ontology for Semantic SOA formalises and extends those sections of the SOA Reference Model described above, as illustrated in Figure 1-1.

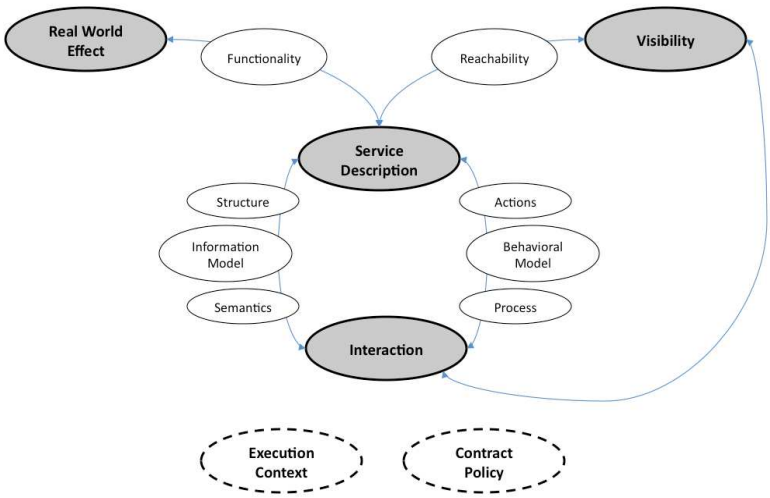


Figure 4-1 – Concepts from SOA-RM as preserved in Reference Ontology [Concept Map]

Oval shapes are used to represent the *top-level* elements from the SOA Reference Model and rectangles represent the subsidiary elements. Those which are shaded are the ones on which we concentrate in the Semantic SOA Reference Ontology. Although *Execution Context* and *Contracting & Policy* are all important issues for SOA, they are less mature from the point of view of ontology-based semantics, and less ready for standardisation. Third party extensions may add these to the Reference Ontology and these may be included in future revisions.

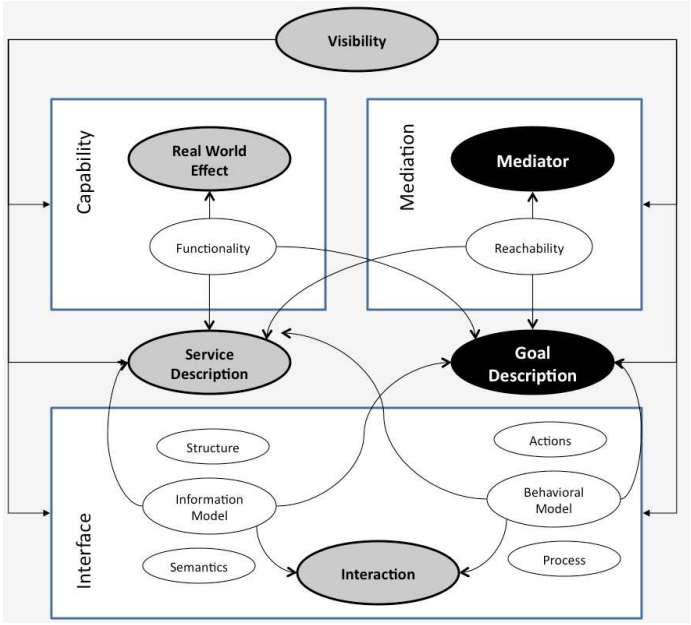


Figure 4-2 - Extension of SOA RM in the Reference Ontology [Concept Map]

In Figure 4-2 we show how we have extended and arranged the Reference Model to enable a thorough semantic description. New elements are shown in solid black. The most notable difference is that *Mediator* takes the place of *Visibility* and *Visibility* becomes are more general concept. *Visibility* is taken as more fundamental to the semantics-driven approach and shown underlying all concepts. Secondly, as well as a *Service Description* we introduce the first class notion of *Goal Description*, which is a top-level element like *Mediator* in our extended model. The purpose of the *Goal Description* is the formal specification of the objective (tasks or activities) that a user wishes to have performed and for which fulfillment is sought. In this way we can make a first class representation of the more restricted sense of *Visibility*, from the SOA RM, and *Reachability* via *Mediator*. The more general concept of *Mediation* is a grouping concept, and represented by a shaded area. In a similar way, we group the description of functionality into a concept *Capability*, and the *Behavioural Model* and *Information Model*, describing *Interaction*, into a concept *Interface*.

The Reference Ontology is introduced in small pieces over the next sections and the complete Reference Ontology can be seen in Figure 4-9.

4.1 Visibility

The two fundamental principles of the semantics-based approach are that: all descriptions of service-oriented concepts should be made in an ontology-based formalism; and that all ontology-based descriptions should be capable of being connected via mediation. For this reason we see visibility, which is the ability to access a description and thereby the service it represents, as the underlying concept of the entire approach. This is a stronger notion of visibility than in SOA RM due to the presence of the Semantic Execution Environment as a proxy; where a semantic description exists, the SEE will execute the semantically-described service on a client's behalf. In the following, we introduce the concepts and requirements for a formalism to be based on ontologies.

4.1.1 Ontologies

Ontologies, as introduced in Section 1.4.2, provide the basis for all elements in the Reference Ontology and contain classes, instances, properties and axioms. The RO reuses the OWL notion of ontology.

4.2 Service Description

SOA RM requires: *"The service description represents the information needed in order to use a service," and states that the definition of service "emphasizes a distinction between a capability that represents some functionality created to address a need and the point of access where that capability is brought to bear in the context of SOA."* In SSOA we regard this as the critical division in the description of a service: the capability description and the interface.

In the Semantic SOA Reference Ontology, these core service descriptions represent a core element in defining Semantic Web Services, which we aim to support automated reasoning over by the use of semantic technologies. Therefore semantic descriptions are associated to all resources, thus services as well. The semantic descriptions are grounded to concrete service interfaces, such that once the semantic description is known, the capability provided by the service can be accessed as well.

It is important to point out that the Semantic SOA Reference Ontology allows for both functional, including behavioral, and non-functional descriptions of the service. While the functional descriptions are formal definitions expressed in terms of ontologies, the non-functional properties are extension of the Dublin Core.

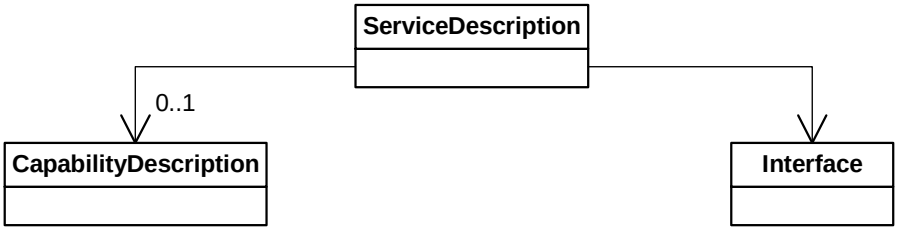


Figure 4-3 - The Top-Level Structure of a Service Description [UML]

4.3 Goal Description

SOA RM defines *awareness* as the state "whereby one party has knowledge of the existence of the other party". Semantic technologies aim to automate as much as possible the process of bringing the service requesters and the services providers in the "awareness state" and to create a dynamic infrastructure able to support all the necessary communication aspects.
Along these lines, the Semantic SOA Reference Ontology has adopted the ontological role separation principle by which the service consumers exist in a specific context, different than the one of the services to be consumed. As a consequence, the requester needs can be independently formalized as *Goals* in accordance with their internal requirements, isolated from the peculiarities of the provider infrastructure, data or behavior models.
Nevertheless, in order to facilitate the matchmaking process between requester goals and provider services, the Reference Ontology defines a GoalDescription as being formed from the same elements as a ServiceDescription: namely a *CapabilityDescription* and a set of *Interfaces*. The CapabilityDescription of a GoalDescription represents the requested capability, i.e. the capability the requester desires to find and consume. The Interface of a GoalDescription describes the interfaces the requester intends to use during the communication with the matching service
.

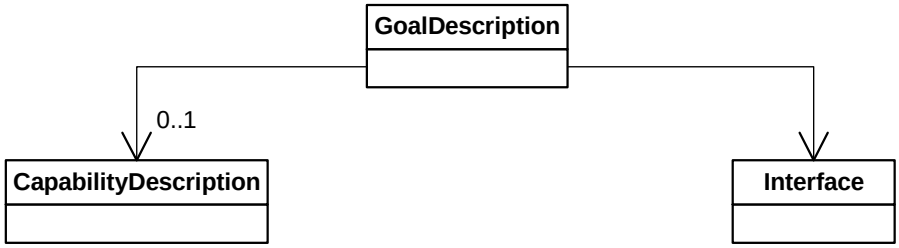


Figure 4-4 - The Top-Level Structure of a Goal Description [UML]

4.4 Capability Description

SOA-RM requires: *"A service description SHOULD unambiguously express the function(s) of the service and the real world effects that result from it being invoked."*

As we have seen in sections 4.2 and 4.3, a CapabilityDescription is a description of the functionality provided by a service or the functionality desired by a service requester and as such can be linked to one or more Service or Goal Descriptions. CapabilityDescriptions are generally used for automating the process of discovering services, by comparing the offered functionality of each provider with the desired functionality of the requester. A Capability is described in terms of conditions on the state of the world that must exist for execution of the service to be possible and conditions on the state of the world that are guaranteed to hold after execution of the service. We make a distinction between the state of the information and the state of the real world, thus these conditions can be broken down into two groups namely those related to the state of the information space (preconditions and postconditions) and those related to the to the state of the real-world (assumptions and effects). By providing these 4 elements, the Reference Ontology allows the state change that occurs in both the information space and in the real world to be effectively described.

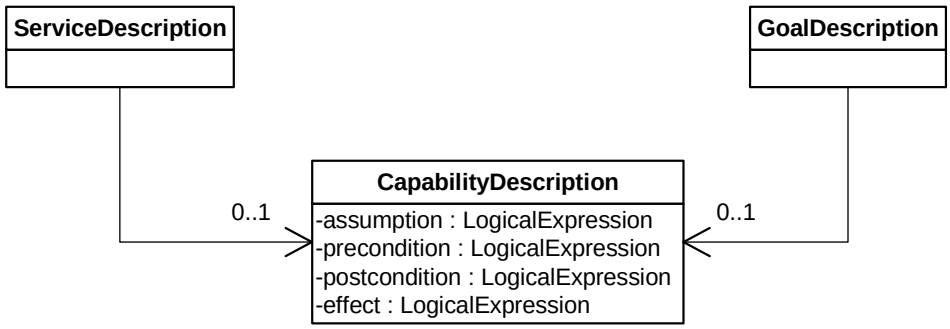


Figure 4-5 – Service and Goal Capabilities [UML]

4.4.1 Functionality

In terms of the SOA-RM the preconditions and postconditions of a service make up the description of its functionality. Preconditions describe the state of the information space prior to execution and postconditions describe the state of the information space after execution. Therefore preconditions can be used to specify what information needs to be available in order for a service to be invoked and postconditions describe what information will be generated by the service into the information space that is communicated back to the consumer.

4.4.2 Real World Effect

Many services that can be invoked will have as the SOA-RM describes a *Real World Effect*, that is that the process of invoking a service will not only change the state of the data sources related to the service requester and service provider but also an actual change will occur to the state of the world, for example when buying a book from a book selling service the physical book will change location from the warehouse to the home of the purchaser. In the Reference Ontology we consider this real world effect by describing the state of the world prior to execution in terms of Assumptions and the state of the world after execution by Effects.

4.5 Interface

SOA-RM specifies that "*the service interface is the means for interacting with a service*". Furthermore, SOA-RM recommends that the interface consists of two parts, Information Model and Behavioral Model. The Information Model is represented both in a semantic and a structural manner.
In the Semantic SOA Reference Ontology the semantic part of information model is based on an ontological description, but this needs to be considered both by the capability and the interface, so this is attached directly to the service (or goal) description, as described in Section 4.5.1. The structural part of the information model needs to be considered only by the communicated information and therefore is represented, via groundings to a schema representation of the appropriate semantic concepts, in the action model, as described in 4.5.2.1.

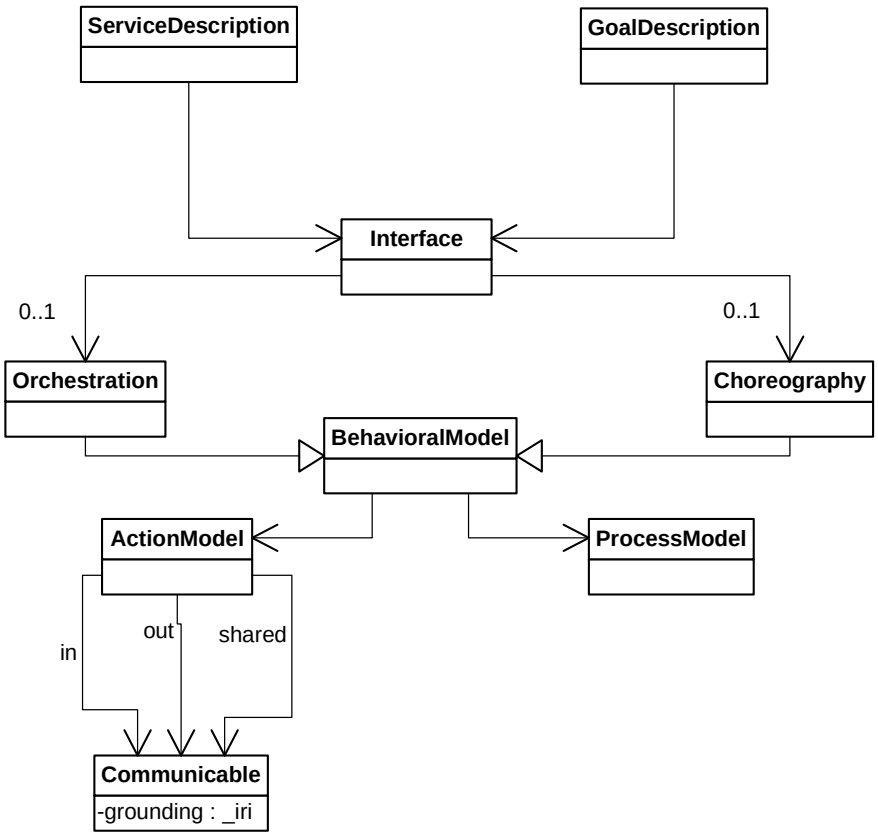


Figure 4-6 - The Structure of an Interface [UML]

For the Semantic SOA Reference Ontology, the notion of behavioural model is specialised into two different concepts, representing different perspectives:

- Service requester perspective - the information that is needed for service execution by the service requester, specified as *Choreography*;
- Communication with other services – information on how the service can coordinate the cooperation between other services in order to fulfill its functionality, specified as the *Orchestration*.

4.5.1 Information Model

"*The information model of a service is a characterization of the information that may be exchanged with the service*". As previously described, for Semantic SOA this information is provided by the domain ontology of the service; this ontology specifies all the information needed for the service execution and for its communication with other services or with the requestors.

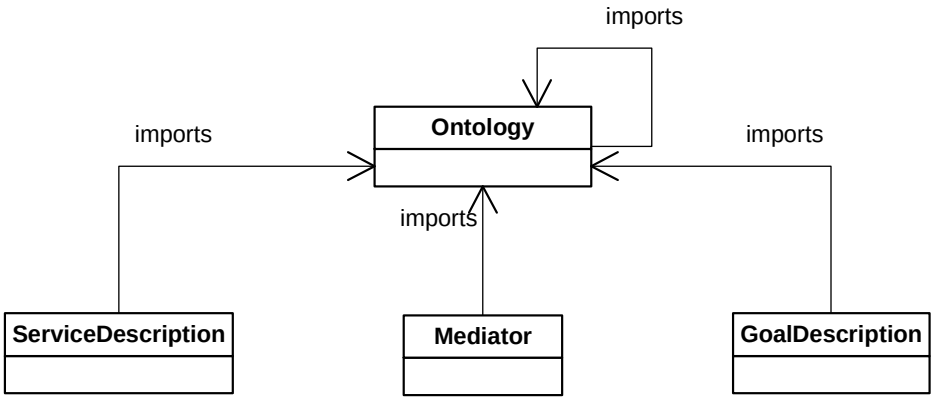


Figure 4-7 Ontologies as Semantic Information Model [UML]

4.5.1.1 Semantics

Whenever parties need to deal with one another, there is a need to have a common understanding of the semantics of the models or messages to be exchanged between them. When the parties use ontologies for describing their information model, this common understanding implies either a previous agreement regarding what ontologies are used, or the existence of a mediator for solving any heterogeneity problems. This will ensure a high degree of automation for the communication.

4.5.1.2 Structure

As described above, some of the concepts (and relations) from the Semantic Information Model will actually be communicated by the service. The structural definition of these components will be represented by the groundings in the Action Model, described in Section 4.5.2.1.

4.5.2 Behavioral Model

The SOA RM defines the Behavioral Model as "*knowledge of the actions invoked against the service and the process or temporal aspects of interacting with the service*". For Semantic SOA this knowledge is encapsulated by the definition of what information needs to be exchanged during the communication, the concepts and relations of an ontology being marked to support a particular role (or mode). Furthermore, the order in which the messages are exchanged needs to be unambiguously specified.

4.5.2.1 Action Model

For specifying what information needs to be exchanged during the communication the concepts and relations of an ontology are marked to support a particular role (or mode). There are five modes defined in the state signature:

- s*tatic* - meaning that the extension of the concept cannot be changed;
- *in* - meaning that the extension of the concept or relation can only be changed by the environment and read by the service;
- *out* - meaning that the extension of the concept or relation can only be changed by the service and read by the environment;
- *shared* - meaning that the extension of the concept or relation can be changed and read by the service and the environment;
- *controlled* - meaning that the extension of the concept is changed and read only by the service.

For using the modes defined in the state signature a grounding mechanism needs to be provided for allowing the environment (i.e. the communication partner) to read or to write information in the services ontology. For each mode except static and controlled, a different grounding mechanism needs to be provided as follows:

- *in* - a **grounding** mechanism for the in items, that implements *write* access for the environment, must be provided;
- *out* - a **grounding** mechanism for the out items, that implements *read* access for the environment, must be provided;
- *shared* - a **grounding** mechanism for the shared items, that implements *read/write* access for the environment and the service, must be provided.

For the static and controlled items a grounding mechanism is not needed, as these items can either be changed only by the service or remain unchanged for the duration of the communication.

4.5.2.2 Process Model

The Semantic SOA Reference Ontology is not prescriptive about what form the behavioural description should take, except that it should take account of the action modes.

OWL-S, for instance, defines a block-oriented process description (composite processes) that coordinate atomic processes which pair and in- and out-action, mapping down to a WSDL operation.

The SOA4All project has introduced a 'minimal service model' where such operations are explicit, and has investigated a resource-oriented model where operations are attached to resource types and bound to the HTTP verbs as an implementation of the REST style. Both of these would be types of process model.

Finally, rules could be used to specify a process model; for instance using the Abstract State Machine formalism employed by WSML, each rule evaluating some conditions on the current state of the service, and prescribing which actions are permissible and how these evolve the service state.

4.6 Mediation

SOA RM defines Visibility as "*the relationship between service consumers and providers that is satisfied when they are able to interact with each other*". Visibility itself subsists in the publication of Service and Goal Descriptions, but a prerequisite of Visibility is represented by Reachability, and when two entities are aware of each other and willing to interact in order to fulfill a need, heterogeneity can be a barrier that prevents this prerequisite to be fulfilled. Given two heterogeneous entities, mediation enables Reachability by resolving mismatches between them, which may include (but is not limited to) a Goal and Service description in the context of discovery.

A mediator is described in terms of the entities it is able to connect and states how it will resolve mismatches. Ontology to Ontology mediators (OO-Mediators) connect ontologies and resolve terminological and representational mismatches, Service Description to Service Description mediators (SS-Mediators) connect service descriptions resolving mismatches between the representation of their functionality and/or in the means by which they are accessed (i.e., between their capabilities and/or interfaces), Goal Description to Goal Description mediators (GG-Mediators) connect Goal descriptions resolving mismatches in the requirements of the service requestor, again either in capability or interface terms, and Service Description to Goal Description (SG-Mediators) connect Service descriptions and goal descriptions, mediating between the consumer's and provider's viewpoint of the functionality and/or its access. Each of these different types of Mediator can specify a mediation service that can offer the functionality for that mediator. By using a Mediation Service, a Mediator explicitly describes the link to a concrete solution to perform mediation. This mechanism allows Mediators to be used to describe pieces of functionality offered by complex services that are able to perform concrete mediation scenarios. A mediation service can either be a Goal or a Service Description. The former links to a Goal that is to be used in the discovery process to find a Service offering the functionality described by the Mediator, while the latter directly links to a Service that is able to offer the functionality described by the Mediator.

By publishing the description of the Mediator and all its needed Ontologies, Goal and Service Descriptions, the requirements for Visibility are met, thus allowing a Goal to interact with the Service.

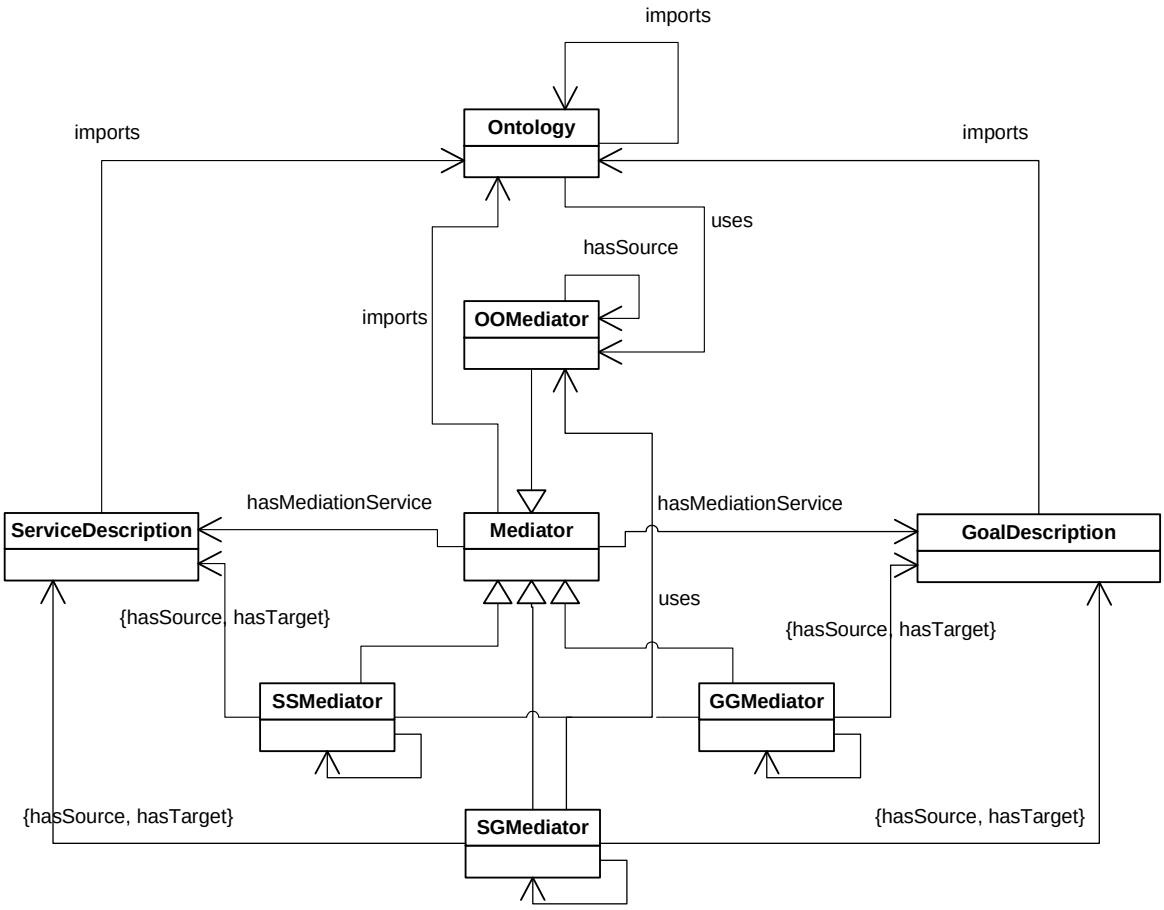
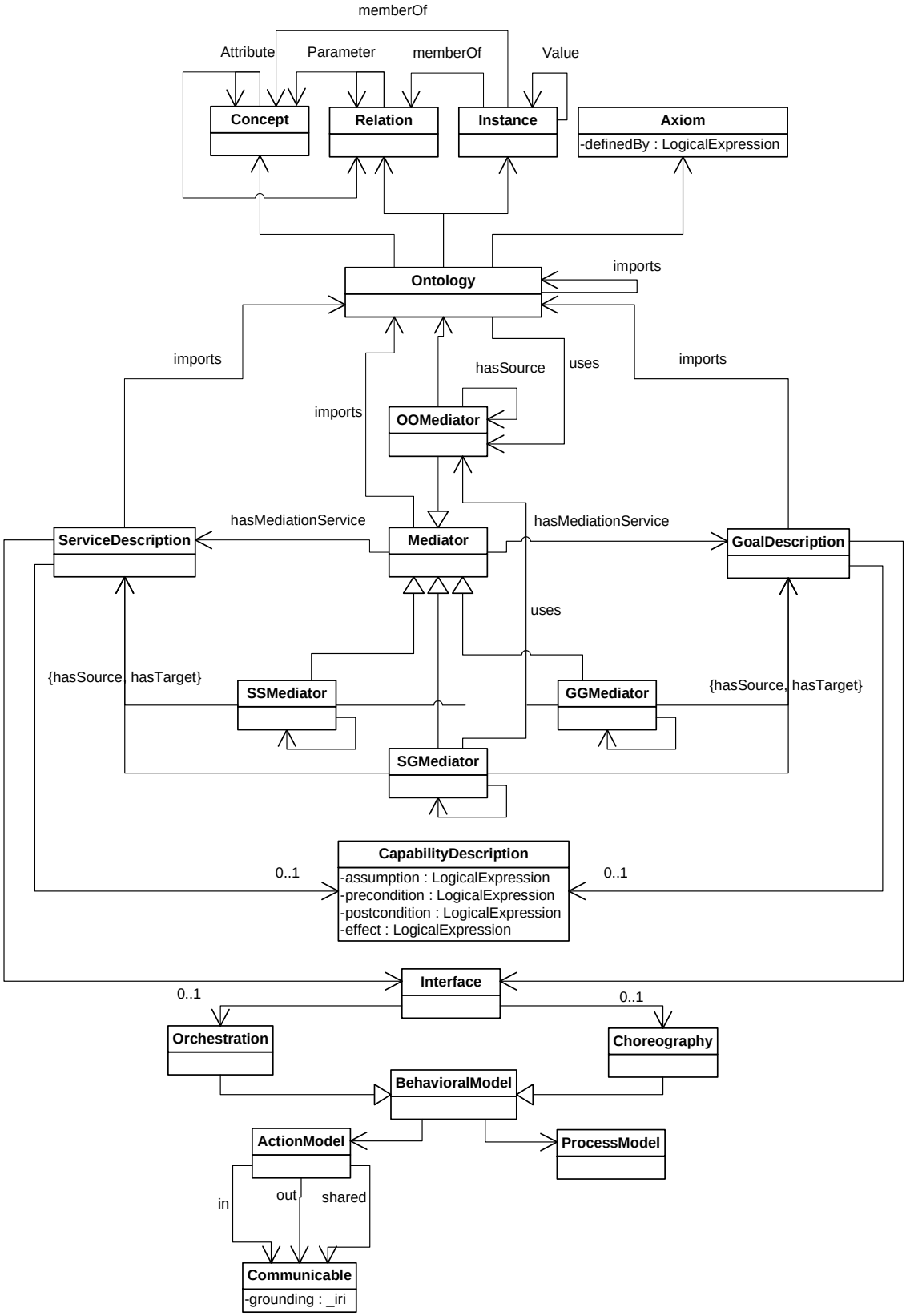


Figure 4-8 – Mediators and their Connection of other RO Concepts [UML]

4.7 Complete Reference Ontology

Figure 4-9 shows complete UML diagram for the Reference Ontology, which combines all the information from **Error! Reference source not found.** to Figure 4-8. The formalization of this ontology in RDFS is presented in Appendix B.



663

664 *Figure 4-9 - The Complete Reference Ontology [UML]*

5 Conformance

This Reference Ontology for Semantic Service Oriented Architectures is an abstract framework for understanding significant entities and relationships between them within a Semantically-enabled Service-Oriented environment. It may be leveraged for the development of related standards or specifications supporting that environment, as well as guiding efforts to realize concrete solutions. As such, it has no explicit conformance statements.

A. Glossary

This section extends the terminology described in Glossary (Appendix A) of the "Reference Model for Service Oriented Architecture, Public Review Draft 1.0" and introduces any new terms needed by the Semantic SOA Reference. The two glossaries are intended to be used together, therefore terms from the other glossary will not be repeated here.

Goal Description-to-Goal Description Mediator (GG-Mediator)

Connects Goal descriptions resolving mismatches in the requirements of the service requestor in terms of the requested functionality and/or in the means by which they wish to access the service

Internet Reasoning Service 3 (IRS III)

A framework and infrastructure that supports the creation of Semantic Web Services according to the WSMO ontology.

Managing End-To-End OpeRations for Semantic Web Services and Processes (METEOR-S)

Project that aims to extend Web service –related standards with Semantic Web technologies to achieve greater dynamism and scalability for Service-oriented Architectures.

Object-oriented Design (OOD)

Object-oriented design is part of OO methodology and it forces programmers to think in terms of objects, rather than procedures, when they plan their code.

Ontology-to-Ontology Mediator (OO-Mediator)

Connects ontology and resolves terminology as well as representation or protocol mismatches.

Resource Description Framework (RDF)

Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata model but which has come to be used as a general method of modeling information, through a variety of syntax formats.

Rule Interchange Format (RIF)

The Rule Interchange Format (RIF) is a W3C recommendation-track effort to develop a format for interchange of rules in rule-based systems on the semantic web. The goal is to create an interchange format for different rule languages and inference engines.

Semantic Annotations for WSDL (SAWSDL)

The Semantic Annotations for WSDL and XML Schema (SAWSDL) W3C Recommendation defines mechanisms using which semantic annotations can be added to WSDL components.

Semantic Execution Environment (SEE)

Execution environment capable to consume semantic messages, discover semantically described Web services, and invoke and compose them for the end-user benefit.

Semantic Web

The **Semantic Web** is an evolving extension of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content. [cite: Wikipedia]

Semantic Service Oriented Architecture (SSOA)

A **Semantic Service Oriented Architecture** (**SSOA**) is a computer architecture that allows for scalable and controlled Enterprise Application Integration solutions. SSOA describes a sophisticated approach to enterprise scale IT infrastructure. It leverages rich, machine-interpretable descriptions of data, services, and processes to enable software agents to autonomously interact to perform critical mission functions. [cite: Wikipedia]

Semantic Web Services (SWS)

Semantic Web Services are self-contained, self-describing, semantically marked-up software resources that can be published, discovered, composed and executed across the Web in a task driven semi-automatic way. Semantic Web Services can be defined as the dynamic part of the semantic web.

Semantic Web Service Ontology (SWSO)

An ontology for Semantic Web Services, which is expressed in two forms: FLOWS, the First-order Logic Ontology for Web services; and ROWS, the Rules Ontology for Web services, produced by a systematic translation of FLOWS axioms into the SWSL-Rules language.

Service-oriented Architecture (SOA)

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed 128 capabilities that may be under the control of different ownership domains.

Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a standardized visual specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model.

Web Ontology Language for Services (OWL-S)

OWL-S is an ontology built on top of Web Ontology Language (OWL) by the DARPA DAML program. It replaces the former DAML-S ontology.

Web Service Description Language (WSDL)

The Web Services Description Language is an XML-based language that provides a model for describing Web services.

Service Description-to-Goal Description Mediator (WG-Mediator)

Connects service descriptions and goal descriptions, mediating between the consumer's and provider's viewpoint of the functionality and/or its access

Service Description-to-Service Description Mediator (WW-Mediator)

Connects service descriptions resolving mismatches between the representation of their functionality and/or in the means by which they are accessed.

762

763 **Web Service Modeling eXecution environment (WSMX)**

764 An execution environment for business application integration where enhanced Web services are
765 integrated for various business applications. It is the reference implementation of WSMO (Web
766 Service Modeling Ontology).

767

768 **Web Service Modeling Language (WSML)**

769 A language that formalizes the Web Service Modeling Ontology (WSMO).

770

771 **Web Service Modeling Ontology (WSMO)**

772 WSMO or Web Service Modeling Ontology is an ontology currently developed to support the
773 deployment and interoperability of Semantic Web Services.

B. RDF(S) Formalization of Reference Ontology

```
@prefix : <http://docs.oasis-open.org/semanticsoa/referenceontology/v1.0#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

:Ontology rdfs:subClassOf owl:Ontology, :Description.

:uses rdfs:domain :Ontology;
      rdfs:range :OOMediator.

:ServiceDescription rdfs:subClassOf :Description.

:GoalDescription rdfs:subClassOf :Description.

:imports rdfs:domain :Description;
         rdfs:range :Ontology.

:has_interface rdfs:domain :Description;
         rdfs:range :Interface.

:offers_capability rdfs:domain :ServiceDescription;
         rdfs:range :CapabilityDescription.

:requires_capability rdfs:domain :ServiceDescription;
         rdfs:range :CapabilityDescription.

:LogicalExpression rdfs:subClassOf rdfs:Literal.

:precondition rdfs:domain :Capability;
              rdfs:range  :LogicalExpression.

:assumption rdfs:domain :Capability;
            rdfs:range  :LogicalExpression.

:postcondition rdfs:domain :Capability;
               rdfs:range  :LogicalExpression.

:effect rdfs:domain :Capability;
        rdfs:range  :LogicalExpression.


:has_choreography rdfs:domain :Interface;
                  rdfs:range  :Choreography.

:has_orchestration rdfs:domain :Interface;
                   rdfs:range  :Orchestration.

:Choreography rdfs:subClassOf :BehaviourModel.

:Orchestration rdfs:subClassOf :BehaviourModel.

:has_actions rdfs:domain :BehaviourModel;
             rdfs:range  :ActionModel.

:has_process rdfs:domain :BehaviourModel;
             rdfs:range  :ProcessModel.

:hasInAction rdfs:domain :ActionModel;
```

```
            rdfs:range :Communicable.

:hasOutAction rdfs:domain :ActionModel;
              rdfs:range :Communicable.

:hasSharedAction rdfs:domain :ActionModel;
                 rdfs:range :Communicable

:Communicable rdfs:subClassOf rdfs:Class.

:grounding rdfs:domain :Communicable;
           rdfs:range rdfs:Literal.

:Mediator rdfs:subClassOf :Description.


:source rdfs:domain :Mediator;
        rdfs:range :Description.

:target rdfs:domain :Mediator;
        rdfs:range :Description.

:mediationService rdfs:domain :Mediator;
                  rdfs:range :Description.


:usesMediator rdfs:domain :SGMediator;
              rdfs:range :OOMediator.

```

Listing 3: Semantic SOA Reference Ontology Expressed in RDF(S) as Turtle

C. Acknowledgements

The chairs of the TC would like to acknowledge the following individuals who where members of the TC during this specification and aided in its completion:

Alessio Carenini, CEFRIEL
Emilia Cimpian, Semantic Technology Institute Innsbruck
Emanuele Della Valle, CEFRIEL
Federico Facca, Semantic Technology Institute Innsbruck
Marc Haines, Individual
Mick Kerrigan, Semantic Technology Institute Innsbruck
Srdjan Komazec, Semantic Technology Institute Innsbruck
Peter Matthews, CA
Matthew Moran, Semantic Technology Institute Innsbruck
Barry Norton, Karlsruhe Institute of Technology
Carlos Pedrinaci, The Open University
Omair Shafiq, Semantic Technology Institute Innsbruck
Maciej Zaremba, Digital Enterprise Research Institute Galway