Reference Ontology for Semantic Service Oriented Architectures

Release Candidate 1, 28 March 2008

Artifact Identifier:

wd-see-semanticsoaro-rc1

Location:

Current: docs.oasis-open.org/ex-semantics/sematicsoaro/latest
This Version: docs.oasis-open.org/ex-semantics/sematicsoaro/0.1
Previous Version: docs.oasis-open.org/ex-semantics/sematicsoaro/0.1

Artifact Type:

semanticsoaro

Technical Committee:

OASIS SEE TC

Chair(s):

John Domingue, Open University, <j.b.domingue@open.ac.uk>
Michal Zaremba, STI, <michal.zaremba@sti2.at>

Editor(s):

Barry Norton, Open University, <b.j.norton@open.ac.uk>
Mick Kerrigan, STI, <mick.kerrigan@sti2.at>
Adrian Mocan, STI, <adrian.mocan@sti2.at>

Contributing Authors:

Emilia Cimpian, STI, <emilia.cimpian@sti2.at>
James Scicluna, STI, <james.scicluna@sti2.at>
Michal Zaremba, STI, <michal.zaremba@sti2.at>
Alessio Carenini, CEFRIEL, <alessio.carenini@cefriel.it>

OASIS Conceptual Model topic area:

SOA

Related work:

Needs to be written

Abstract:

Needs to be written

Status:

This document is currently a working draft and will be update as necessary to bring it up to public review draft status in the coming weeks/months. Please send all comments to the editors.

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 www.oasis-open.org/committees/semantic-ex.

Notices

Table of Contents

1 Introduction

Although Service Oriented Architectures (SOAs) have gathered more attention within Business Organizations, for a long time there was still no clear understanding of what an SOA in fact is. SOA was consequently defined in the SOA Reference Model [1] . However, with the emerging **Semantic Web** technologies, in particular **Semantic Web Services (SWSs)**, new breeds of SOAs are being developed: **Semantic Service Oriented Architectures (SSOA)**. SSOA use semantic technologies to further solve problems that SOAs are limited by. They provide a means to further automate important SOA features, such as discovery, composition and interoperability of and between services.

Different SSOAs are currently being developed in the research community, which have common features to one other. The purpose of this document is thus to define a common reference model for SSOAs. This model will be defined formally using an ontology. Thus this reference ontology will serve as a reference point for different implementations of SSOAs.

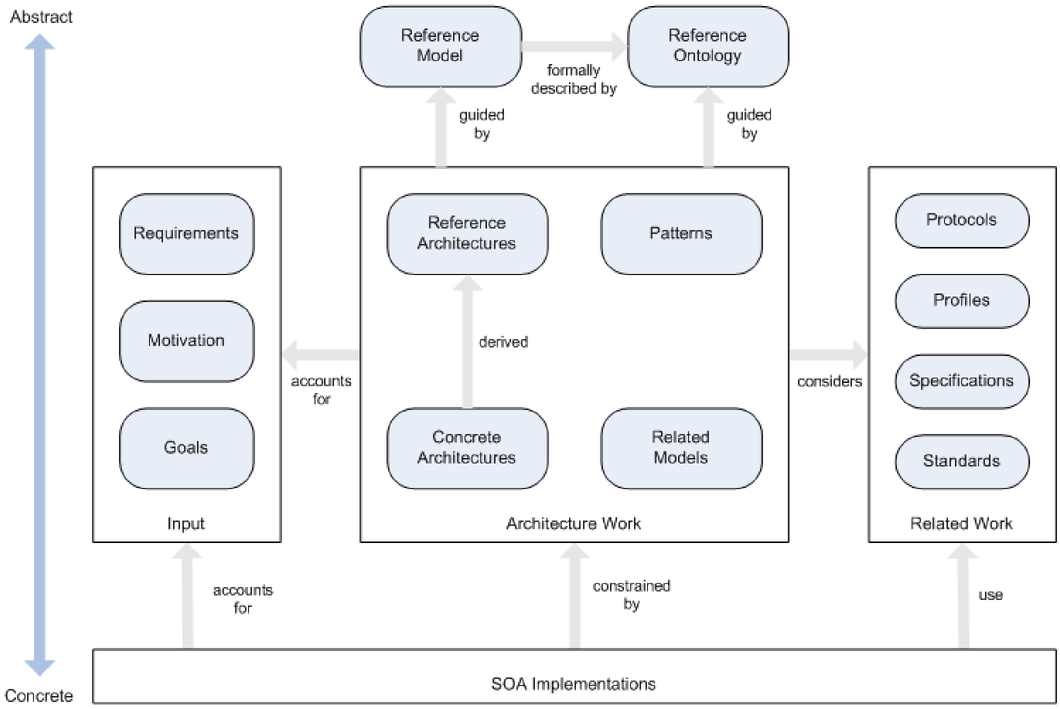


Figure 1-1 - The Reference Ontology and how it relates to other work

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. Our Reference Ontology 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 we have started work, the SOA-RM committee have also started work on a Reference Architecture, but we shall take this to mean our own Semantic SOA Reference Architecture, and Concrete Architectures refer to implementations of semantics-enabled SOAs such as WSMX [2] , IRS III [3] and METEOR-S [4] . The Related Models include the Web Service Modeling Ontology (WSMO) [5] , Semantic Annotations for WSDL (SA-WSDL) [6] the Web Ontology Language for Services (OWL-S) [7] and the Semantic Web Services Ontology (SWSO) [8] .

As for plain SOA, Patterns define more specific categories for SSOA designs. The Protocols and Profiles (those considered as part of the related work) are the same as for classical SOAs. However, with respect to Specifications and Standards, we further take into account emerging Semantic Web Languages such as WSML, RDF, OWL, RIF and SWSL. These de-facto “standards” 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 emphasize on automation, as stated earlier.

1.1 Motivation and Scope

Why introduce Semantics? What are Semantics anyway? With the term “Semantic” we mean the formal (and thus unambiguous) description of some particular object (more in Section 2). Within our context, 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 objectives above.

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 the Service-oriented paradigm;
- Standards architects and analysts developing specifications that rely on Service Oriented Architecture concepts;
- Decision makers seeking a "consistent and common" understanding of Service Oriented Architectures;
- Users who need a better understanding of the concepts and benefits of Service Oriented Architectures;
- 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.

This section 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. This section 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). This section 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 WSML in Appendix B. Note that any other Ontology language (e.g. OWL) can be used to define such an Ontology. We chose WSML since it provides an easy to use syntax and provides different language variants for different types of logical expressivity.

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

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL that appear in this document are to be interpreted as described in [RFC2119 – need reference].

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 herein, no detailed information can be derived from the concept maps.

Figure 1-2 - A basic 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 graphic describes the nature of each relationship.

1.4.2 Ontologies

Within the body text of this document we use UML Class Diagrams to illustrate the ontology. The formal definitions are however made in WSML. 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; secondly we need a language that allows us to attach elements of this model to SWS elements, including goals, and WSML is the only language that allows this.

Specifically, this document sticks to the ontology definition facilities of WSML. 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). The Execution Scenarios will attach Reference Ontology concepts, and Reference Architecture goals, to *service* descriptions 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 WSML descriptions. In the following section we reproduce these definitions.

Concepts

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 WSML concepts. Where the namespace into which concepts 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.

```
concept A

concept _”http://www.example.com/ontologies/ns1#B”
```

Listing 1: Example Concepts in WSML

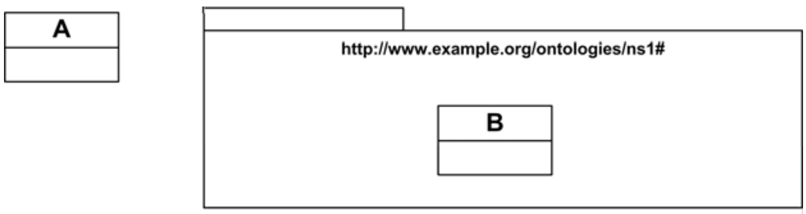


Figure 1-3: Representation of WSML Example Concepts in UML Class Diagram

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. Regarding the representation of attributes of WSML concepts, see below.

Subsumption

The fundamental relationship between concepts in WSML 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.

```
concept A

concept B subConceptOf A

concept C

concept D subConceptOf {A, C}
```

Listing 2: Example of Subsumption between Concepts in WSML

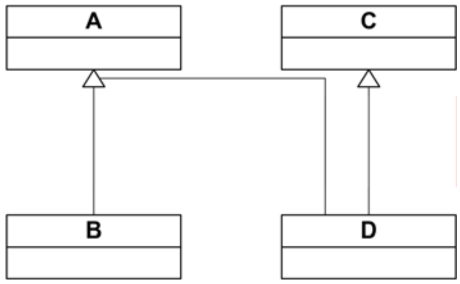


Figure 1-4: Representation of Subsumption Example in UML Class Diagram

Attributes

The other explicit relationship between concepts in WSML is via *attributes*. 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 concept whose definition includes the attribute, and the other side the attribute range. The name of the association will be the name of the attribute; where the attribute name is the default ‘hasA’, where ‘a’ is the name of the concept that is the attribute range, we shall often omit this. Cardinality constraints are represented, where possible, by a constraint on the association. Figure 1-5 hence corresponds with Listing 3.

```
concept E

concept F
   hasE ofType (0, 1) E

concept G
  hasEorF ofType EorF

concept EorF

axiom anEisEorF definedBy
   ?e memberOf E implies
   ?e memberOf EorF.

axiom anFisEorF definedBy
   ?f memberOf F implies
   ?f memberOf EorF.
```

Listing 3: Example of Attributes between WSML Concepts

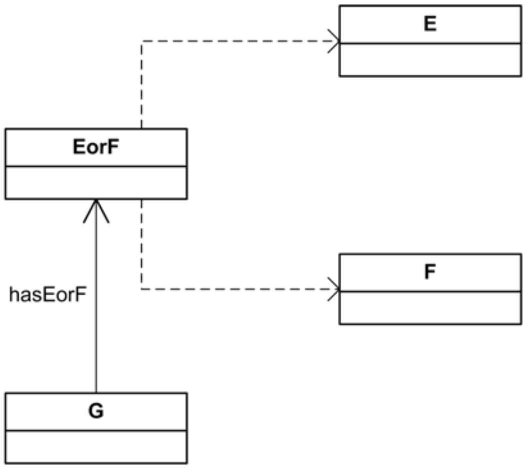


Figure 1-5: Representation of Attributes Example in UML Class Diagram

We also make use of disjunctive attribute ranges by way of an intentionally-defined union class, as shown by attEorH of concept G.

2 Semantics and SOA

As introduced in the Reference Model for Service Oriented Architecture (SOA-RM) committee specification, the notion of Service Oriented Architecture has received a lot of attention in the software design and development community. Service Oriented Architectures provides an architectural mechanism for building applications from unassociated units of functionality called services that have no calls to one another embedded within them. In other words SOA is an architecture that enables an application developer to build an application from loosely coupled services, allowing applications to respond more quickly to changes in market conditions and improving the reusability, modularity, composability and interoperability of functionality that an engineer develops when building an application.

Sadly building Service Oriented Architectures using existing services involves large amounts of human effort in the process of finding and using these services. This human effort is due to the fact that standards for describing services, for example the Web Service Description Language (WSDL), are purely syntactic in nature and thus no automated support for finding and using pre-existing services can be created. When building an application using SOA the engineer is looking for Web services that are available, either within his company's repository of services or on the Web at large that can fulfill a given piece of functionality. Each time the engineer identifies a location where a service invocation is required he must find candidate services that can fill this slot by browsing in UDDI and ebXML repositories. As these repositories are syntactic in nature the engineer will perform keyword matches against the services available in the repository and select candidates by reading the textual descriptions provided in these repositories, if there are any. Having selected some candidates the engineer must obtain the associated WSDL documents for each of the Web services and begin the process of understanding the endpoints that are made available by each service in terms of the functionality they perform, the inputs that they expect and the outputs that the provide. The engineer may need to get in contact with the providers of the Web service to clarify the functionality offered by the service or perform test invocations against the service to check the behavior of the service. Finally the engineer will make a selection of one or more services that can fulfill the job and add them to his application.

Not only is this process human intensive, but the solution that arises from it is not exactly the adaptable decoupled architecture that Service Oriented Architectures promise. Imagine the scenario where a new service comes on the market after the engineer has selected and integrated candidate services into the application. This new service has better functionality than existing services and is also available at a lower price. This service will never be available to the application, and thus to the end-users of the application, unless the engineer finds the service, interprets its function, and integrates it into the application. A similar scenario involves the case where the selected service(s) for a given piece of core functionality within the application are not available due to being overloaded, offline for maintenance or are discontinued. Essentially the application as a whole will not function until the engineer has found and integrated an alternate Web service for this functionality.

2.1 Semantics

The main limitation of SOA as mentioned above is that the standards that are used for describing Web services are purely syntactic in nature and thus large amounts of human effort are required to perform tasks like finding services; But what is the alternative to syntactic descriptions? Semantics is the study of meaning and a semantic description offers the opportunity of providing an unambiguous mechanism for describing things. Semantics comes in many forms, some of which may already be familiar to you. Very light forms of semantics include annotations or tags that can be placed on an entity in order to give a semantic 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. Of course the semantics of these annotations is very light and to bring more semantic meaning to the annotations being used 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 term *banana* is sub class of the term *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, them 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 nre information to be inferred for 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

Semantic Web Services are the extension of ontologies to describe Web services in such a way that 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 Web service that currently has a syntactic description in the form of a WSDL document will also have a semantic description in some formalism once it becomes a Semantic Web Service, in this way it can be seen that Semantic Web Services are not a reinvention of Web services but an enhancement to them. In order to effectively describe Web 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 modelled in order to effectively describe Web services semantically and build Semantically Enabled Service-oriented Architectures.

Once Web services are described semantically it allows for many of the tasks performed by the engineer in building and maintaining and application using SOA to 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 the Service Oriented Architecture, now extended with semantic descriptions to create a Semantically Enabled Service-oriented Architecture (SESA), to dynamically bind to services at run time, removing the hard wired behavior that we see in current applications. When new services appear on the market that fulfill functionality needed by the application, they will 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 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 in 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 SOAs the OASIS SOA-RM Technical Committee[1] proposes an abstract framework for understanding the main entities and the relationships between them within a services 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*.

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 a prescribed interface*.
- A service is *opaque to the service consumer* except from the information and behavioral models in the interface and the information required to asses if a service suits the requester needs.
- *Consequences of invoking a service* should be either response information to the invocation or a change to the shared state of the defined interface.

It is important to not that SOA-RM makes a clear distinction between the capability of a service (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, it identifies three fundamental concepts related with dynamics of the service: *Visibility, Interaction* and *Real World Effect* (see Figure 3-1).

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

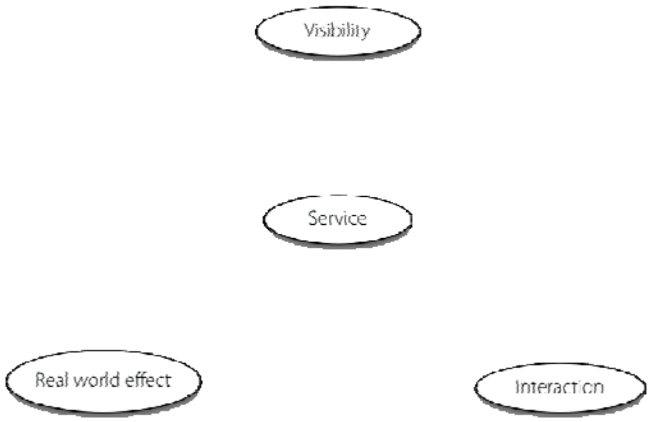


Figure 3-1. Fundamental Concepts of Service Dynamics (from [1])

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 public directory for example.
- *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.

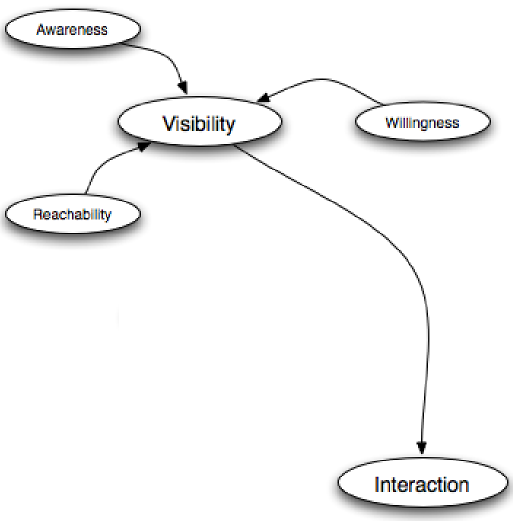


Figure 3-2. Service Visibility (adapted from [1])

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 (see Figure 3-3):

- *Information Model* of a service characterizes the information that may be exchanged with the services and only descriptions of data and information that can be potentially exchanged with the service are included in the information model. The information model can be also portioned in:
    - o *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 implied effects of 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.

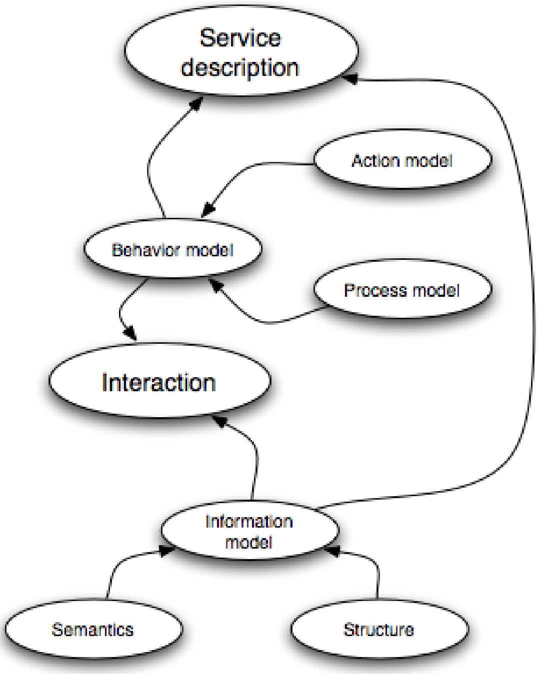


Figure 3-3. Service Interaction (adapted from [1])

The r*eal world effect* it 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 the interaction of the 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:

- That the service is reachable or not.
- That the function the service provides is the function required by the requester
- The set of policies the services operates under.

- That the service complies with the service consumer's policies.
- How 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 interface.

- *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 form it being invoked. This piece of information should be expressed in a general-enough way to be understandable by service consumers while in 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 humans 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.

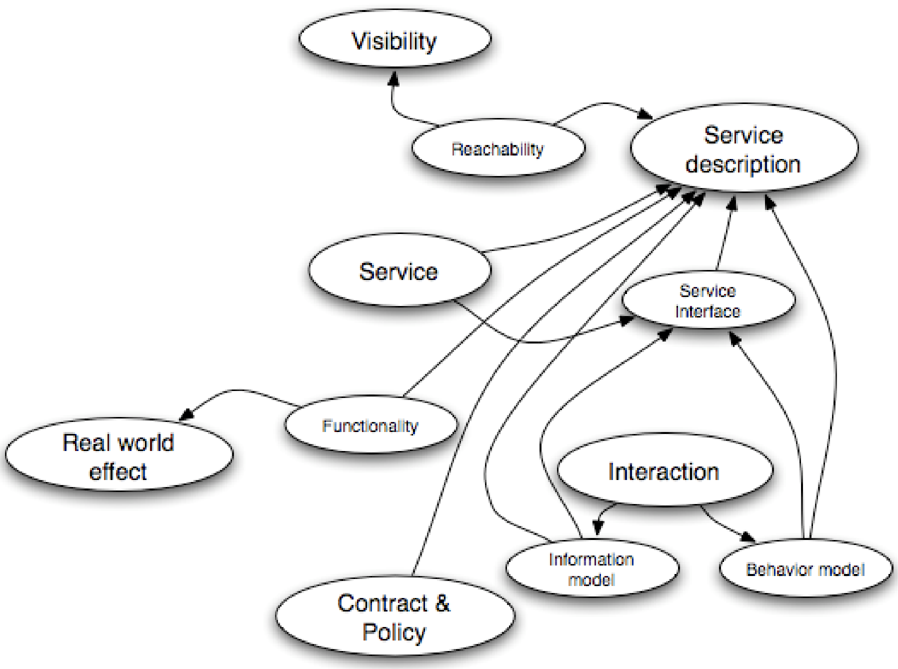


Figure 3-4. Service Description (from [1])

The *service policy* represents the constraints or the conditions on the use, deployment or description of a service while a *contract* is a measurable assertion that governs the requirements and expectations of one or more parties. Policies potentially apply to various aspects of SOA such as security, manageability, privacy, etc. but they could also apply 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.

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

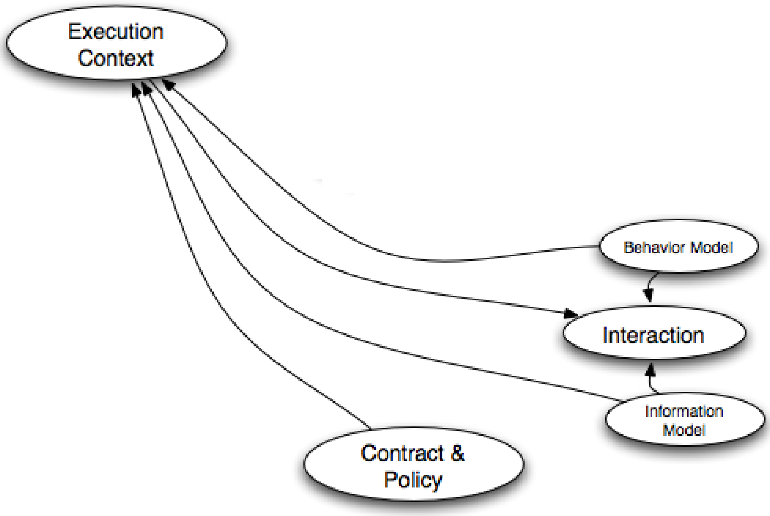


Figure 3-5. Execution Context (adapted from [1])

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.

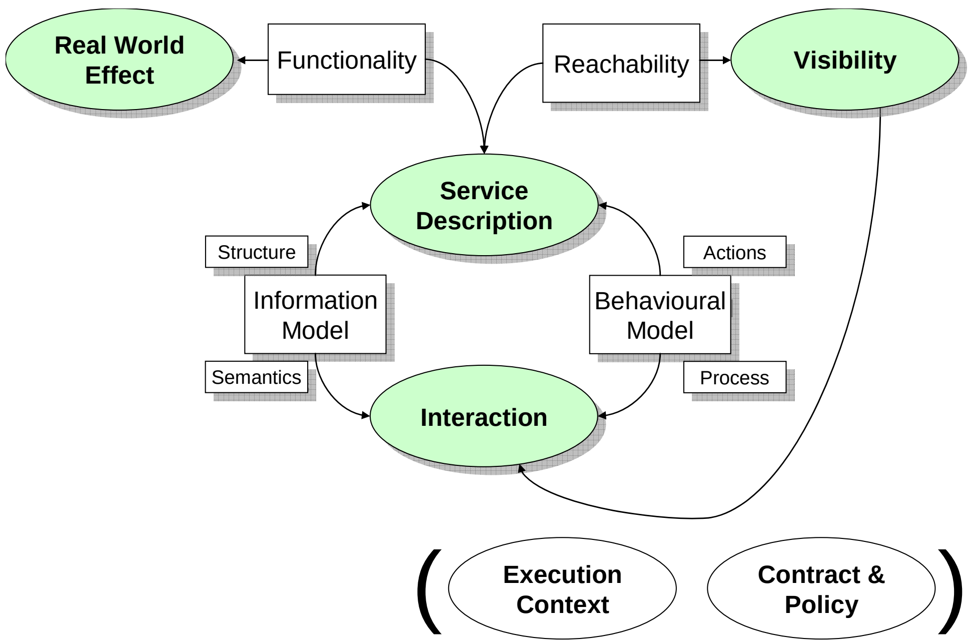


Figure 4-1 - Reference Ontology Basis from Reference Model

Oval shapes are used to represent the top-level elements from the SOA Reference Model, rectangles the others, and those which are shaded are the ones on which we concentrate in the Semantic SOA Reference Ontology. Although Execution Context and Contracting and Policy are all important issues for SOA, they are less mature and ready for standardisation.

In Figure 4-2 we show how we have extended and arranged the Reference Model to enable a thorough semantic description. The most notable difference is that we replace the Visibilty concept with the concept of Mediator. 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. Goal Description is a formal description of the requirements for a service *from the point of view of a consumer*. 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 and Information models, describing Interaction, into a concept Interface.

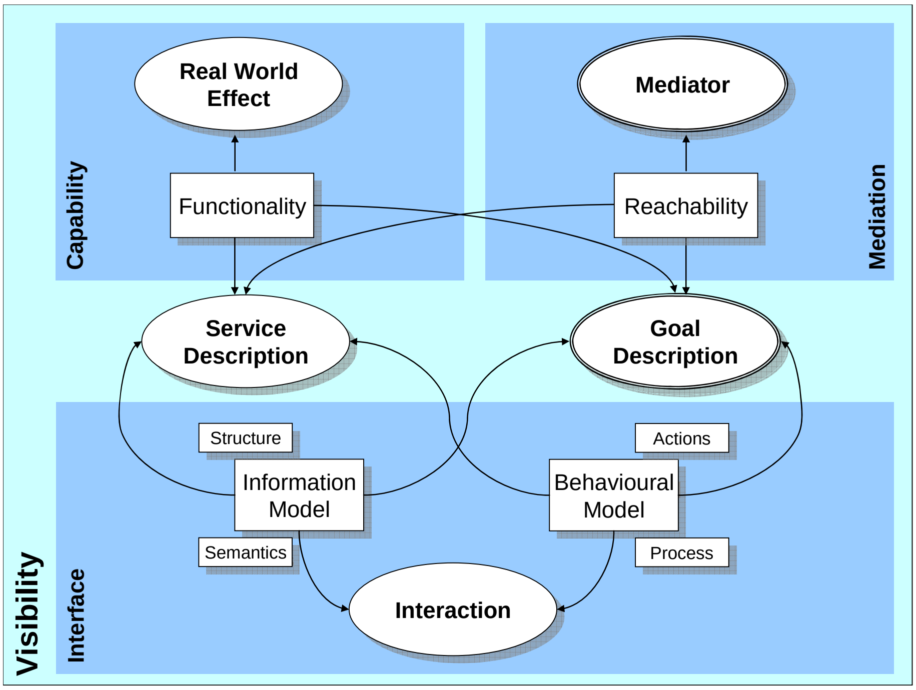


Figure 4-2 - Reference Ontology as Extension of Reference Model

The Reference Ontology is introduced in small pieces over the next sections and the complete Reference Ontology can be seen in Figure 4-10.

4.1 Visibility

The two fundamental principles of the semantics-based approach are that: all descriptions of services-oriented concepts should be made in an ontology-based formalism; 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. 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 Concepts, Instances and Axioms. Service Descriptions, Goal Descriptions, and Mediators can import Ontologies in order to utilize the terminology that they provide.

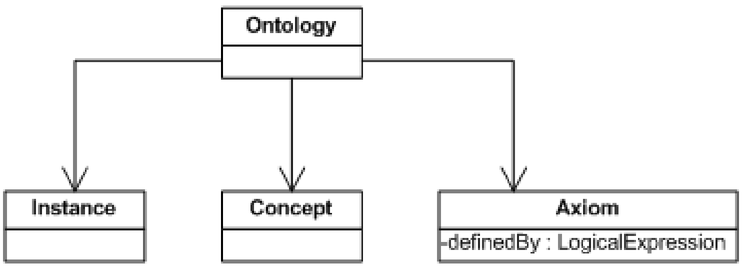


Figure 4-3 - Ontologies and their Contents

4.1.2 Concepts

Concepts provide a means for describing pieces of terminology and the can be related to each other via the subclass-superclass relationship (see Subsumption in Section 1.4.2). Concepts also have attributes that allow other relationships between classes to be captured.

4.1.3 Instances

Instances are identifiable or anonymous members of concepts and provide values to the attributes of those concepts. Instances may be explicitly declared as members of concepts of they may be implicit via axioms.

4.1.4 Axioms and Logical Expressions

Axioms define logical expressions that must hold over all contents of their containing ontology in order for this to be consistent. These can be used to support an explicit style of modelling, where instances and their concept memberships are declared explicitly and axioms merely constrain their allowed membership and attribute values (cf. relational database constraints), or intentionally, where concepts may be implicitly populated via axioms.

4.2 Service Description

SOA RM requires: *"The service description represents the information needed in order to use a service."*

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 realizations, such as once the semantic description is known the implementation of the service can be found 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, and might contain human-readable descriptions as well.

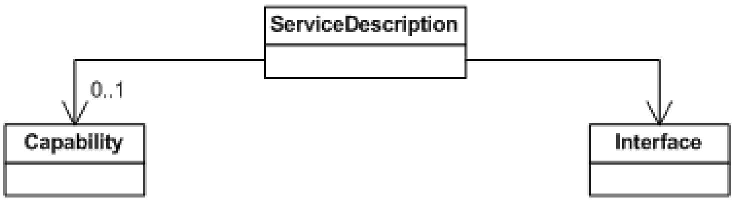


Figure 4-4 - The Top-Level Structure of a Service Description

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 that 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: a *Capability* and an *Interface*. The Capability 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.

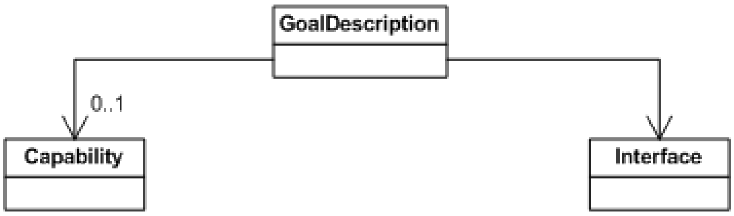


Figure 4-5 - The Top-Level Structure of a Goal Description

4.4 Capability

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 Capability 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. Capabailities are generally used for automating the process of discovering services, by comparing the offered functionality of each provider with the desired funcitionality 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 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.

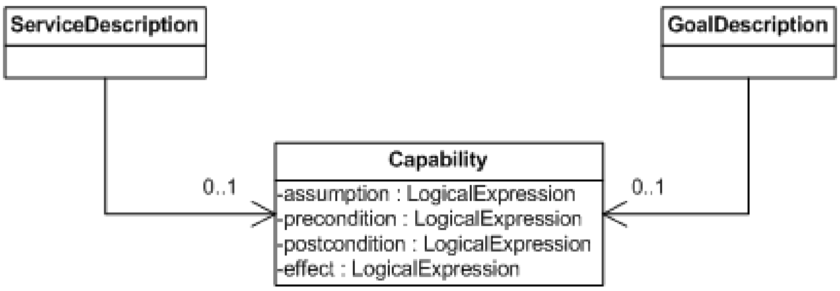


Figure 4-6 – Service and Goal Capabilities

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 exectution. 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.

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 servoce 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 staee of the world after execution by Effects.

4.5 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.

A mediator is described in terms of the entities it is able to connect and states how it will resolve mismatches. Namely, OO-Mediators connect ontologies and resolve terminology as well as representation and protocol mismatches, while WG-Mediators connect Services and Goals. 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.

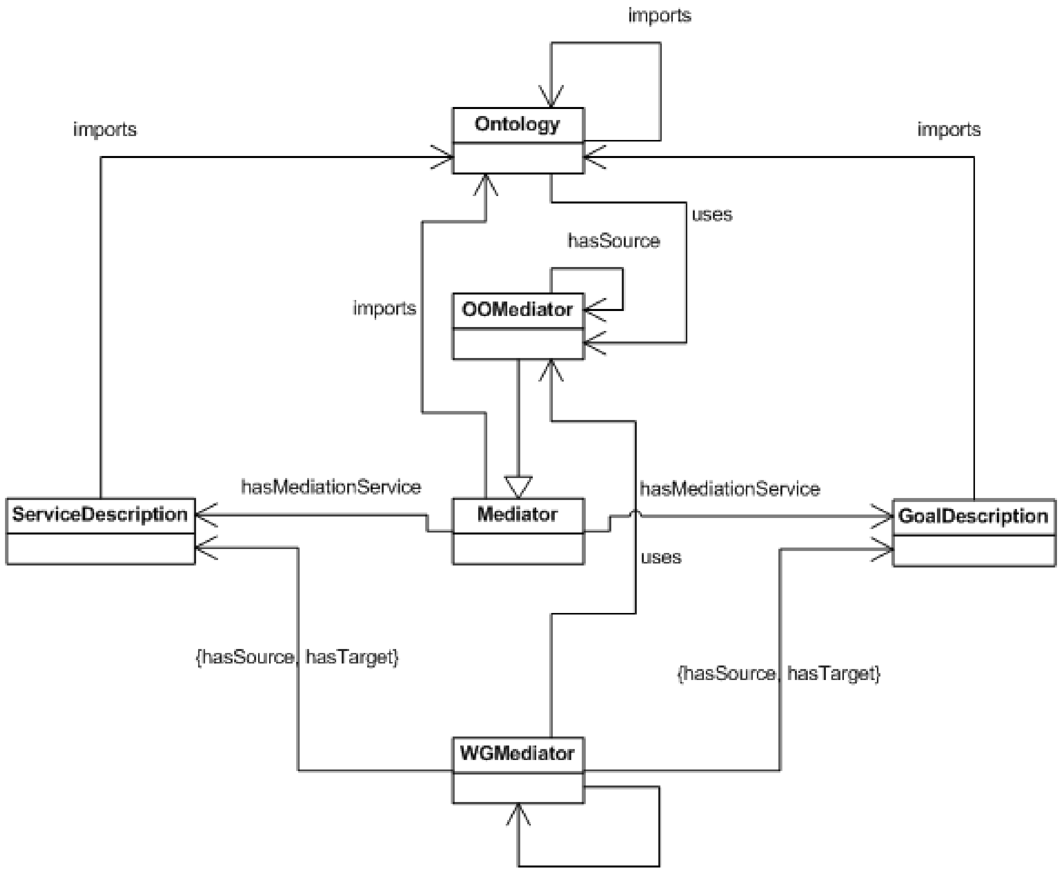


Figure 4-7 – Mediators and their Connection of other RO Concepts

4.6 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, and their roles will be described in the following subsections.

For the Semantic SOA reference Ontology the service interface is also an important part of the ervice description. It specifies in detail how the communication with the service should take place, from two different perspectives: 1) the invoker perspective – what information is needed for the service execution specified as Choreography, and 2) communication with other services – that is, how the service can coordinate the cooperation between other services in order to fulfill its functionality, specify as the *Orchestration*.

The Service Interface encapsulates all the information from the Information and Behavioral Model, providing a clear and concise description of the information and communication pattern needed for interacting with the service from the invoker's perspective.

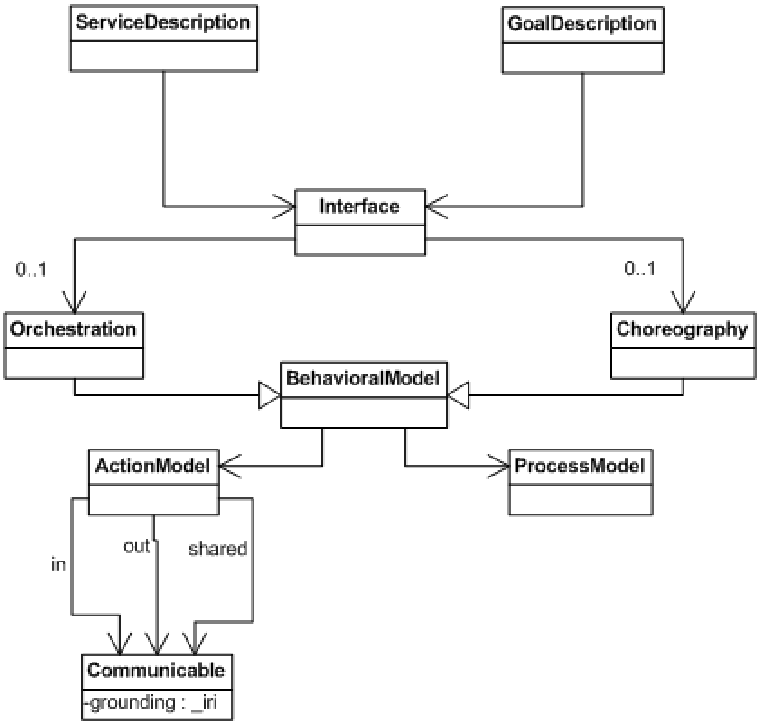


Figure 4-8 - The Structure of an Interface

4.6.1 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.

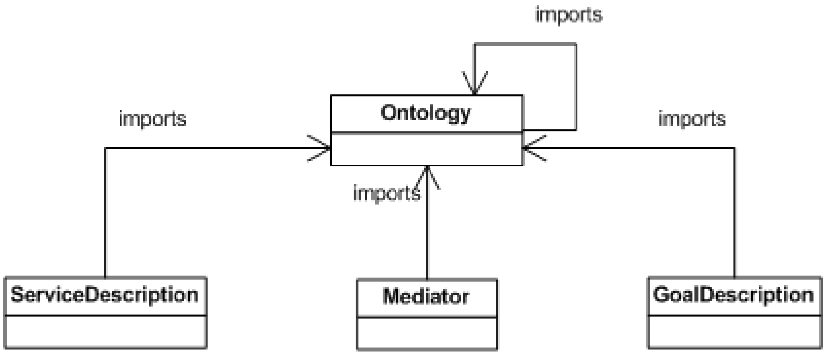


Figure 4-9 Ontologies as Information Model

4.6.1.1 Structure

The information model of a service has to have a given structure, a standard form of the required information in order to ensure the successful invocation of the service. This structure is given by the domain ontology, which prescribes the format of the information needed or provided by the service.

Section 1.4.2, presents the format of the ontologies; the information model is described (like any other entity presented in this document) in terms of this ontologies

4.6.1.2 Semantics

The parties involved in a communication need to have a common understanding of the semantic of the exchanged messages. 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 automaticity for the communication.

4.6.2 Behavioural Model

The SOA RM defines the Behavioural 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.

4.6.2.1 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, namely *static*, *in*, *out shared* and *controlled*: *static* - 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.

4.6.2.2 Process Model

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.

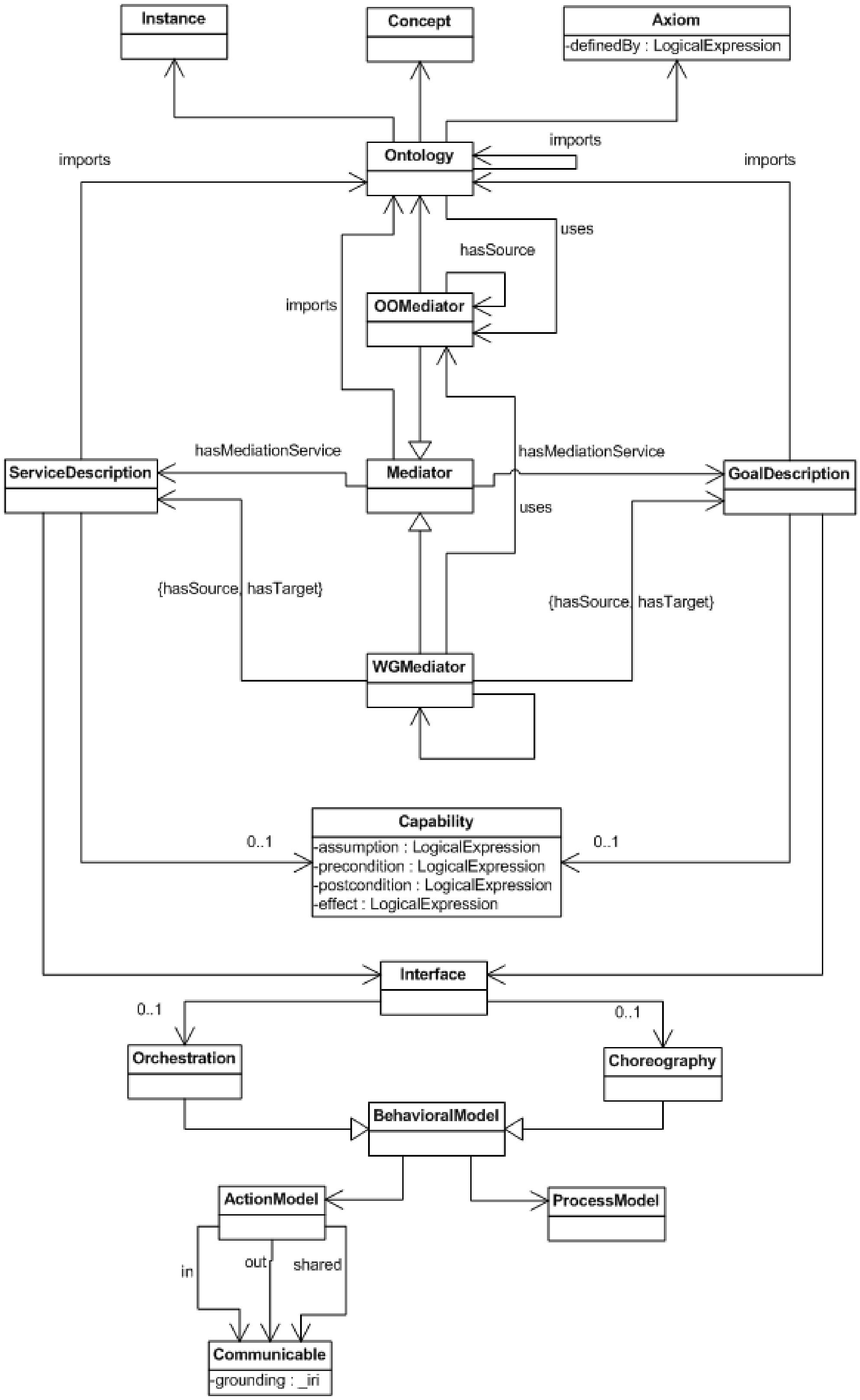
- *out* - a **grounding** mechanism for the out items, that implements *read* access for the environment, must be provided.
- *shared* - a **grounding** mechanism for the shared items, that implements *read/write* access for the environment and the service, must be provided .

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

Furthermore, a set of transition rules are needed for defining the order in which the messages can be exchanged. These rules can be specified using the Abstract State Machine methodology, each rule evaluating some conditions on the current state of the service, and prescribing what activities should be performed if the conditions are fulfilled.

4.7 Complete Reference Ontology

In Figure 4-10 shows complete UML diagram for the Reference Ontology, which combines all the information from Figure 4-3 to Figure 4-9. The formalisation of this ontology in WSML is presented in Appendix B.



612

613 *Figure 4-10 - The Complete Reference Ontology*

5 References

5.1 Normative

Normative references go here

5.2 Non-Normative

Non-Normative references go here

[1] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz (eds.): Reference Model for Service Oriented Architecture 1.0, OASIS SOA-RM Technical Committee Specification, 2 August, 2006, available at: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[2] A. Haller, E. Cimpian, A. Mocan, E. Oren, C. Bussler: WSMX: A Semantic Service-Oreinted 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, C. Pedrinaci: IRS-III: A Broker-based Approach to Semantic Web Services. In Journal of Web Semantics (to appear) 2008.

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

[7] 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/

[8] 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/

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.

Semantic Service Oriented Architectures

Definition

Semantic Web

Definition

Semantic Web Services (SWS)

Definition

B. WSML Formalisation of Reference Ontology

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
namespace { _"http://www.semantic-soa.org/ReferenceOntology#",
              RO _"http://www.semantic-soa.org/ReferenceOntology#"
 }

ontology _"http://www.semantic-soa.org/ReferenceOntology#"

concept Ontology
  imports ofType Ontology
  hasConcept ofType Concept
  hasInstance ofType Instance
  hasAxion ofType Axiom
  uses ofType OOMediator

concept Concept
  hasInstance ofType Instance

concept Instance

concept Axiom
  hasLogicalExpression ofType _"http://www.wsmo.org/wsml/wsml-
syntax#logicalExpression"

concept ServiceDescription
  imports ofType Ontology
  offersCapability ofType (0 1) Capability
  hasInterface ofType Interface

concept GoalDescription
  imports ofType Ontology
  requiresCapability ofType (0 1) Capability
  hasInterface ofType Interface

concept Capability
  hasPrecondition ofType _"http://www.wsmo.org/wsml/wsml-
syntax#logicalExpression"
  hasAssumption ofType _"http://www.wsmo.org/wsml/wsml-
syntax#logicalExpression"
  hasPostcondition ofType _"http://www.wsmo.org/wsml/wsml-
syntax#logicalExpression"
  hasEffect ofType _"http://www.wsmo.org/wsml/wsml-
syntax#logicalExpression"

concept Interface
  hasChoreography ofType (0 1) Choreography
  hasOrchestration ofType (0 1) Orchestration

concept Choreography subConceptOf BehaviourModel

concept Orchestration subConceptOf BehaviourModel

concept BehaviourModel
```

```
  hasActionModel ofType (1) ActionModel
  hasProcessModel ofType (0 1) ProcessModel

concept ActionModel
  hasInAction ofType (1) Communicable
  hasOutAction ofType (1) Communicable
  hasSharedAction ofType (1) Communicable

concept Communicable
  grounding ofType (0 1) _iri

concept MediationService

axiom aServiceIsAPotentialMediationService definedBy
  ?m memberOf ServiceDescription implies
  ?m memberOf MediationService.

axiom aGoalIsAPotentialMediationService definedBy
  ?m memberOf GoalDescription implies
  ?m memberOf MediationService.

concept Mediator
  imports ofType Ontology
  hasMediationService ofType (0 1) MediationService


concept WGMediator subConceptOf Mediator
  hasSource ofType (1) WGMediatorSource
  hasTarget ofType (1) WGMediatorTarget
  RO#usesMediator ofType (1) OOMediator

concept WGMediatorSource

axiom aServiceIsAPotentialWGMediatorSource definedBy
  ?x memberOf ServiceDescription
  implies
  ?x memberOf WGMediatorSource.

axiom aGoalIsAPotentialWGMediatorSource definedBy
  ?x memberOf GoalDescription
  implies
  ?x memberOf WGMediatorSource.

axiom aWGMediatorIsAPotentialWGMediatorSource definedBy
  ?x memberOf WGMediator
  implies
  ?x memberOf WGMediatorSource.

concept WGMediatorTarget

axiom aServiceIsAPotentialWGMediatorTarget definedBy
  ?x memberOf ServiceDescription
  implies
  ?x memberOf WGMediatorTarget.

axiom aGoalIsAPotentialWGMediatorTarget definedBy
  ?x memberOf GoalDescription
  implies
```

```
  ?x memberOf WGMediatorTarget.

axiom aWGMediatorIsAPotentialWGMediatorTarget definedBy
  ?x memberOf WGMediator
  implies
  ?x memberOf WGMediatorTarget.

concept OOMediator subConceptOf Mediator
  hasSource ofType OOMediatorSource

concept OOMediatorSource

axiom anOntologyIsAPotentialOOMediatorSource definedBy
  ?x memberOf Ontology
  implies
  ?x memberOf OOMediatorSource.

axiom anOOMediatorIsAPotentialOOMediatorSource definedBy
  ?x memberOf OOMediator
  implies
  ?x memberOf OOMediatorSource.
```

Listing 4: Semantic SOA Reference Ontology Expressed in WSML

C. Acknowledgements

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

Participants:

Marc Adlam, Oracle Corporation
Zachary Alexander, Individual Member
Michael Altenhofen, SAP AG
Anuprlya Ankolekar, Institute of Applied Informatics and Formal Description Methods (AIFB)
Tim Banks, IBM
Charlton Barreto, Adobe Systems
David Brock, MW2 Consulting
Dario Cerizza, CEFRIEL
Joseph Chiusano, Booz Allen Hamilton
Emilia Cimpian, Digital Enterprise Research Institute (DERI)
David Cunningham, Booz Allen Hamilton
Emanuele Della Valle, CEFRIEL
Paul Denning, Mitre Corporation
Marin Dimitrov, Ontotext Lab/Sirma Group
John Domingue **(chair)**, The Open University
Elmar Dorner, SAP AG
Matthew Dovey, Oxford University
Mike Evanoff, ManTech Enterprise Integration Center (e-IC)
Dieter Fensel, Digital Enterprise Research Institute (DERI)
Sri Gopalan, Booz Allen Hamilton
Peter Gratzer, Sun Microsystems
Stephen Green, Individual Member
Marc Haines, Individual Member
Thomas Haselwanter, Digital Enterprise Research Institute (DERI)
Graham Hench, Digital Enterprise Research Institute (DERI)
Hyun Jung, Korean National Computerization Agency
Mick Kerrigan **(secretary)**, Digital Enterprise Research Institute (DERI)
Eunju Kim, Korean National Computerization Agency
Hong-Gee Kim, Digital Enterprise Research Institute (DERI)
Paavo Kotinurmi, Digital Enterprise Research Institute (DERI)
Ho Kyoung Lee, Korean National Computerization Agency
Jean-Luc LEVESQUE, EdiEyes
Sandeep Maripuri, Booz Allen Hamilton
Monica Martin, Sun Microsystems
Tom Michaud, Software AG, Inc.
Dugki Min, Individual Member
Adrian Mocan, Digital Enterprise Research Institute (DERI)
Matthew Moran, Digital Enterprise Research Institute (DERI)
Barry Norton, The Open University
Srinivas Padmanabhuni, Infosys Technologies
Yuanying Pan, Changfeng Open Standards Platform Software Alliance
Ash Parikh, Raining Data Corporation
Koustubh Pawar, CA
Carlos Pedrinaci, The Open University
Jebastin Prabaharan, Infravio, Inc.
Jiyong Pyon, Korean National Computerization Agency
Brahmananda Sapkota, Digital Enterprise Research Institute (DERI)
Svante Schubert, Sun Microsystems
Ron Schuldt, Lockheed Martin

Omair Shafiq, Digital Enterprise Research Institute (DERI)
Clifford Thompson, Individual Member
Walt Truszkowski, NASA
Laurentiu Vasiliu, Digital Enterprise Research Institute (DERI)
Tomas Vitvar, Digital Enterprise Research Institute (DERI)
Alexander Wahler, NIWA-Web Solutions
Michal Zaremba **(chair)**, Digital Enterprise Research Institute (DERI)
Maciej Zaremba, Digital Enterprise Research Institute (DERI)

D. Revision History

[optional; should not be included in OASIS Standards]

Rev	Date	By Whom	What
wd-00	2007-09-13	Mick Kerrigan	Initial TOC from F2F Meeting
wd-01	2007-09-21	Adrian Mocan	Content added to Section 3
wd-02	2007-09-21	Barry Norton	Content added to Section 4
wd-03	2007-10-21	Barry Norton	Content added to Sections 1.4 and 4 Added Appendix B
wd-04	2007-10-21	James Scicluna	Updated Introduction
wd-05	2007-10-21	Barry Norton	Updated Section 4 (diagrams), introduce new Section 4.1 on Visibility
wd-09	2008-01-15	Emilia Cimpian	The interface descriptions added
wd-10	2008-01-16	Mick Kerrigan	The ontology and capability description added
wd-11	2008-01-16	Adrian Mocan	The goal description added
wd-12	2008-02-14	Barry Norton	Edited Section 4.1 and WSML Appendix and changed references to this
Wd-13	2008-03-10	Adrian Mocan	Figure 3-1 Fundamental Concepts of the Service Dynamics added.
Wd-14	2008-03-21	Mick Kerrigan	Removed Conclusions, Updated Introduction, and cleaned up document
Wd-15	2008-03-26	Alessio Carenini	Added draft for Section 4.5 (Mediation)
Wd-16	2008-03-26	Barry Norton	Accepted all changes, corrected references, reworded introduction, started to highlight glossary terms, further changes to Section 1.4.2 and 4.5
WD-17	2008-03-27	Mick Kerrigan	Updated UML Diagrams in section 1 and minor English changes to mediation section.
RC1	2008-03-28	Mick Kerrigan	Created Release Candidate 1