



D3.1v01. WSMO Primer

DERI Working Draft 19 April 2004

This version:

<http://www.wsmo.org/2004/d3/d3.1/v0.1/20040419>

Latest version:

<http://www.wsmo.org/2004/d3/d3.1/v0.1>

Previous version:

<http://www.wsmo.org/2004/d3/d3.1/v0.1/20040328>

Editors:

Sinuhe Arroyo
Michael Stollberg

Authors:

Sinuhe Arroyo
Michael Stollberg
Ying Ding

This document is also available in non-normative [PDF](#) version.

Copyright © 2004 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Abstract

The Web Service Modeling Ontology (WSMO) is an ontology for describing various aspects related to Semantic Web Services. The primer is designed to provide the reader with the fundamental knowledge required to effectively use WSMO. It presents the basic concepts of WSMO in regard to [Version 0.2](#) of the language specification. It describes how to model goals and Web Services, further it shows how to model the different types of mediators introduced in the specification, together with the role played by ontologies in WSMO. Finally, it describes the content and purpose of other documents in the specification.

Related Documents

Currently, the primer refers to the following deliverables within the WSMO specification:

- WSMO Specification: [D2v02. Web Service Modeling Ontology - Standard](#)
- WSMO Use Case Modeling and Testing: [D3.2v01. WSMO Use Case Modeling](#)

and Testing

Table of contents

- 1. Introduction**
 - 2. Modeling Services**
 - 2.1 Basic Concepts and Model
 - 2.2 WSMO Principles
 - 3. Use case detailed description**
 - 4. WSMO Modeling Elements**
 - 4.1 Non-Functional Properties
 - 4.2 Ontologies
 - 4.3 Goals
 - 4.3.1 Goal Non-functional Properties
 - 4.3.2 Post-conditions
 - 4.3.3 Effects
 - 4.3.4 Used Mediators
 - 4.4 Web Services
 - 4.4.1 Non-functional properties
 - 4.4.2 Used Mediators
 - 4.4.3 Capability
 - 4.5 Mediators
 - 5. WSMO Specification Structure**
 - 6. References**
 - Acknowledgments**
-

1. Introduction

The Web Service Modeling Ontology (WSMO) is a formal ontology and language for describing the various aspects related to Semantic Web Services. WSMO takes a layered approach that starts with a basic WSMO, called WSMO-Lite, continues with a mature set of concepts called WSMO-Standard, and ends with a full ontology called WSMO-Full, as shown in figure 1.

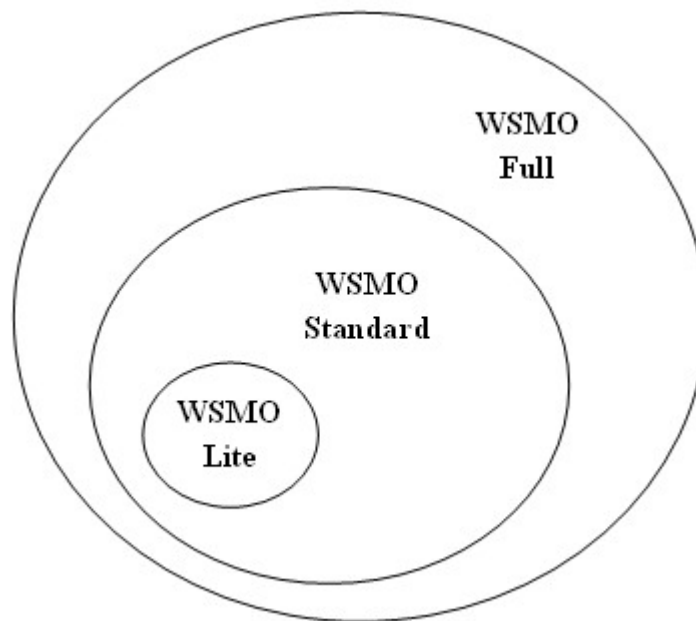


Figure 1. WSMO-Lite, WSMO-Standard, WSMO-Full

The WSMO language is suitable for modeling Web Services, aiming at providing the facilities required by its usage process namely, discovery, invocation, composition, execution, monitoring, mediation and compensation.

The objective of WSMO and its surrounding efforts is to define a coherent technology for Semantic Web Services. Means for semi-automated discovery, composition, and execution of Web Services shall be based on logical inference-mechanisms, because of the well-known competences of suchlike techniques, and its appropriateness for the purpose. In order to allow suitable logic-based reasoning on Semantic Web Service, the description language has to provide reasonable expressiveness for describing relevant aspects of the Services, together with well defined formal semantics that support effective reasoning. WSMO applies F-Logic as the underlying logic formalism, since it provides a standard model theory, it is a full first order logic language, provides second order syntax while staying in the first order logic semantics, and it has a minimal model semantics. A further benefit of F-Logic is that implemented inference engines are already available.

The primer is based on the work carried on [D2v02. Web Service Modeling Ontology - Standard](#) (2004-03-06) final draft, which sets the pillars of the Web Service Modeling Ontology (WSMO), presenting its core elements and the relation among them. The WSMO specification and in particular the Web Service Modeling Ontology - Standard takes as starting point the Web Service Modeling Framework (WSMF) [[Fensel & Bussler, 2002](#)] further refining and extending concepts.

It is the principal aim of the primer to provide an introduction to WSMO - Standard, describing it with a great deal of detail, and showing how to model Goals, Mediators and Semantic Web Services; further, it exemplifies the use of ontologies. The target audience is researches, software engineers and developers, besides anyone dealing with Semantic Web Services. It is the Primer intention to answer question such as:

- What is WSMO in a nutshell?
- How does it look like?

- How are Semantic Web Services modeled using WSMO?
- How WSMO complements current Web standards?
- How does it improve current Semantic Web Services initiatives?

It is not the primer intention to provide a WSMO specification, for such purpose a close look should be taken into the rest of the documents ([working drafts and final drafts](#)) which currently are part of the project.

In order to introduce the basic concepts and model, a flight ticket buying example will be used through out the Primer. In such example an agent tries to buy a plane ticket to travel from Innsbruck (Austria) to Galway (Ireland).

For the shake of simplicity, and in order to better show the concepts detailed in [D2v02. Web Service Modeling Ontology - Standard](#) (2004-03-06) final draft, it is assumed that one Web Service fulfills all the functionality required through out the process of buying a plane ticket, while this, of course might not be the case in many systems, where a service might be invoked to search for a ticket, another to reserve it, and probably third one to buy and pay.

The WSMO Primer is organized as follows: [Section 2](#) introduces the basic concepts and model behind WSMO, together with its principles; [Section 3](#) describes in detail the use case that guides the primer; [Section 4](#) provides a detailed overview and understanding on the main elements that compose the WSMO specification, this section is divided in five subsections, [Section 4.1](#) shows how to model non-functional properties; [Section 4.2](#) exemplifies the use of ontologies, together with the essential components used to define them; [Section 4.3](#) overviews goals and how they are used to specify Web Services objectives; [Section 4.4](#) presents Web Services, and [Section 4.5](#) exemplifies the use of mediators, which provide the means to bypass interoperability differences among the rest of the elements that compose the specification. [Section 5](#) presents the WSMO specification, while [Section 6](#) provides some useful references.

2. Modeling Services

WSMO is intended to provide an efficient and simple way to describe the various aspects of Semantic Web Services with the aim of providing a solution to its semi-automated usage and further develop a solution for the Enterprise Application Integration problem. This section describes the approach and basic ideas behind WSMO in order to tackle these problems.

2.1 Basic Concepts and Model

WSMO defines the modeling elements for describing several aspects of Semantic Web Services. The conceptual grounding of WSMO is based on the Web Service Modeling Framework (WSMF) [[Fensel & Bussler, 2002](#)], wherein four main components are defined, as shown in figure 2.

- **Ontologies** provide the formal semantics to the information used by all other components.
- **Goals** specify objectives that a client may have when consulting a Web Service.
- **Web Services** represent the functional part which must be semantically described in order to allow its semi-automated use.
- **Mediators** used as connectors provide interoperability facilities among the rest

of components.

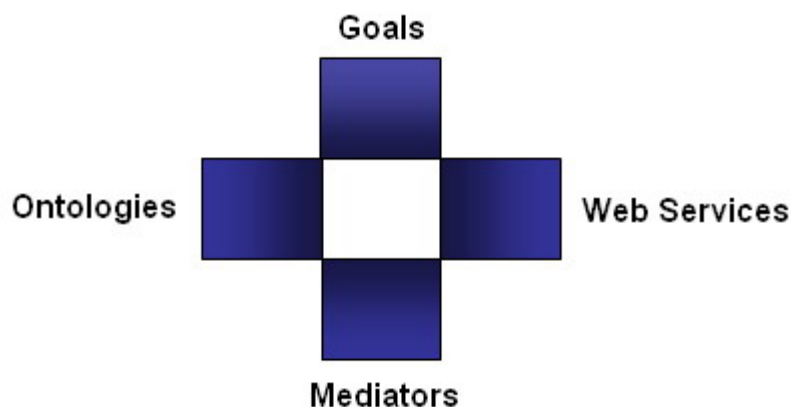


Figure 2. WSMO Components

Ontologies represent the key element in WSMO. They serve a twofold purpose, first they define the information formal semantics, and second allow to link machine and human terminologies. They play a key role in the WSMO specification since, they give meaning to the different elements, providing the terminology that can be aligned to allow its interoperability from a semantic point of view. WSMO specifies the following constituents as part of the description of an ontology: non-functional properties, used mediators, axioms, concepts, relations and instances.

Ontologies are used to: (1) express goals in a machine processable and understandable language, i.e buy a plane ticket; (2) they permit to enhance Web Services so they can be matched against goals, i.e Web Services provided by on-line travel agencies; and (3) interconnect the different elements with each other by means of mediators, i.e services and goals using different similar terminology. A more detailed Ontology description is given in section [4.2 Ontologies](#).

Goals represent the objectives to be fulfilled when consulting a Web Service. The WSMO definition of goal is restricted to post-condition, effects, non-functional properties and used mediators, leaving aside pre-conditions and assumptions. A goal is conceptualize as buying a plane ticket, in our concrete example. Goals provide the means to express high level description of a concrete task. For a more detail look at goals refer to section [4.3 Goals](#).

Web Services provide the functional behavior that is to be described, in order to allow its discovery, invocation, composition, execution, monitoring, mediation and compensation if required. They represent the atomic piece of functionality that can be reused to build more complex ones. In this sense it is straight forward to assimilate Web Services to procedures, methods or functions of any programming paradigm, with the advantage of its natural platform independence and distribution over the Web. In the WSMO specification Web Service are described, by means of capability, interfaces, used mediators and non functional properties. A Service is describe by one and only one capability, contrary to interface, where a Service can have multiple interfaces.

In respect to the plane ticket example, Web Services provide the search, book, buy and pay functionality (all in one service in this particular case), respectively expressed in the associated Web Service capability. For more detail information about how to model Web Services within WSMO please refer to section [4.4 Web Services](#).

Mediators allow to overcome functional and non-functional mismatches among the different elements that build the WSMO specification. Currently the specification counts with four different types of mediators, which are classified in two main classes: *refiners* and *bridges*. While refiners are used to define new components as a specialization of an existing one, bridges help to overcome interoperability problems by enabling components to interact with each other. Among the four types of mediators that currently build the specification, ggMediators and ooMediators are refiners, while wgMediators and wwMediators are counted as bridges. In the general case WSMO defines mediators by means of non-functional properties, source component target component and mediation service, where source and target component can be a mediator, a Web Service, and ontology or a goal. In our particular example, mediators could be easily used to arbitrate among goals and the capabilities of a given Web Service, probably expressed using different ontology languages and terminology. Such would be the case of ooMediators as part of a wgMediator. More detail about the particular use of mediators and the role they play in WSMO will be provided in section [4.5 Mediators](#).

2.2 WSMO Principles

Every ontology design is based on principles that can be abstracted from the domain that is going to formally represent. In the following the principles of the WSMO are outlined in order to provide a guide to the ontology design decisions.

- **Decoupling and Mediation.** In a e-commerce environment applications should be as disaggregated as possible, hiding internal business intelligence from public access and carrying communication among processes by means of public message exchange protocols. In order to achieve this objectives the decoupling of the different components that build an e-commerce application is a must.
Scalable communications should allow anybody to speak with everybody, in order to fulfill this n:m communication style, terminologies should be aligned and interaction styles should be intervened. To better achieve this principles, a mediation approach, which allows to map different business logics, together with the ability to establish the difference between public and private processes of e-Services, should be taken.
- **Interface vs. Implementation.** When formally describing interacting entities it is important to recognize the differences among the interface of an interacting entity, its internal implementation as well as its externally visible and internal behavior. The WSMO specification focus on the external visible behavior only.
- **Peer-to-peer vs. Client/Server.** Interacting entities, sometimes called agents, are characterized in general to one entity interacting with one or more other entities. In the simplest case, two entities are interacting. In the context of Semantic Web Services, the interaction is between equal partners, i.e., one entity sending a message and the other one receiving the message are equal in their level of control over the other entity. This is called a peer-to-peer approach in contrast to a client/server approach, where the client drives the interaction as the controlling entity. The WSMO shall recognize the fact that all entities are equal in their control structure leading in the general case to conversations amongst equal partners.

- **Execution semantics.** The WSMO is a representation of the domain of Semantic Web Services. Since it formally describes the interactions of agents a formal execution semantics has to be defined in order to uniquely specify the execution behavior at runtime. The concepts of WSMO shall have a formal execution semantics to ensure a consistent execution model.

3. Use case detailed description

Through out this document examples in regard to a buy plane ticket use case are presented. In a nutshell the use case can be resumed in the following sentence '*buy plane ticket to fly from Innsbruck (Austria) to Galway (Ireland)*'. To make more concrete the problem the Web Service used to book and buy our flight should take under consideration the following premises, namely: (1) the total travel time including plane connecting time, if required, should be smaller than 7 hours; (2) the traveling date is on March 29th, 2004 and the ticket is a single way one; (3) the most expensive ticket that we are willing to pay should not cost more than 600 euros; (4) there are no restrictions about the company, as long as the ticket is the cheapest possible one; (5) the whole trip should not have more than two jumps, this is, we are willing to change planes once, but only once; (6) in case there are no direct flights from Innsbruck to Galway, our preferred intermediate destinations are Dublin, Amsterdam, Frankfurt or London; (7) the time between connecting flights should be smaller than 2 hours; and finally, (8) we would like to flight tourist class.

4. WSMO Modeling Elements

The following section explains the WSMO Modeling Elements, therein describing its position inside the WSMO.

4.1 Non-Functional Properties

Each element of WSMO (ontologies, goals, services and mediators) counts among its definition an instance of the so-called "Non Functional Properties". Non-functional properties specify information regarding the name of the item, authorship, copy rights, etc. WSMO uses [Dublin Core Meta Data Set](#) as reference to define non-functional properties. They are defined as a set of tuples, consisting on an attribute and its value. Non-functional properties are used to describe aspects that do not affect functional aspects of the element. Currently they are classified in two main categories: core properties and Web Service specific properties. [Listing 1](#) provides an example of core properties, while in section [4.4.1 Non-functional properties](#) the same can be found for a Web Service.

The following provides and example on non-functional properties for an ontology element. More concretely for the ontology presented in [Listing 2](#).

```
ONFP-1: [
    title ->> {Plane Itinerary Ontology}
    creator ->> {Sinuhe Arroyo}
    subject ->> {travel, leisure, plane}
    description ->> {this ontology defines the
```

```
ontological notions needed to describe plane itineraries}
  publisher ->> {DERI - Innsbruck}
  contributor ->> {DERI - Galway}
  date ->> {2004-03-29}
  type ->> {DataSet}
  format ->> {text plain}
  identifier ->> {onfp-1}
  source ->> {http://www.wsmo.org/2004/d3/d3.1/v0.1}
  language ->> {English}
  relation ->> {ontology-01}
  coverage ->> {Europe}
  rights ->> {DERI liability}
  version ->> {0.1}
]: nonFunctionalProperties
```

Listing 1. Non Functional Properties

The attribute Type has been defined following the [Dublin Core Meta Data Set](#) recommended best practice for encoding type values as defined in the DCMI Type Vocabulary [[DCMI](#)], which includes among others the selected type. Any other type defined in DCMI Type Vocabulary [[DCMI](#)] is also considered best practice for defining types within the WSMO specification.

The attribute Format has been defined following the [Dublin Core Meta Data Set](#) recommended best practice for encoding format values as defined in the Internet Media Types [[MIME](#)], which includes among others the selected type. Any other type defined in the Internet Media Types [[MIME](#)] is also considered best practice for defining formats within the WSMO specification.

The attribute Date has been defined following the [Dublin Core Meta Data Set](#) recommended best practice for encoding date values as defined in ISO 8601 [[W3CDTF](#)], which includes among others the selected format. Any other format defined in ISO 8601 [[W3CDTF](#)] is also considered best practice for defining dates within the WSMO specification.

The attribute coverage has been defined following the [Dublin Core Meta Data Set](#) recommended best practice for encoding spatial locations. A value has been selected from the Thesaurus of Geographic Names [[TGN](#)]. The Thesaurus of Geographic Names [[TGN](#)] is also considered best practice for defining spatial locations within the WSMO specification. For temporal periods or jurisdiction best practice is to select a name from a controlled vocabulary.

The attribute version has been defined following the [Unicode Standard](#) recommended best practice for encoding version numbers.

4.2 Ontologies

Ontologies are commonly understood as an "formal explicit specification of a shared conceptualization" [[Gruber, 1993](#)]. In WSMO, ontologies provide the machine-readable semantics of the information used by all other components, thus allowing interoperability and information interchange among components. An ontology is described by **concepts**, **relations**, **axioms** and **instances**, conforming to the elements of ontologies as defined in [Open Knowledge Base Connectivity](#). Concepts

are defined by their subsumption hierarchy and their attributes including range specification. Relations describe links between concepts, which carry information on parameters. Axioms are specified as F-Logic expressions and help to formalize domain specific knowledge. Instances are concrete individuals of concepts, that carry concrete values for attributes and relations of the corresponding concept.

In regard to our *'buy plane ticket to fly from Innsbruck (Austria) to Galway (Ireland)'* case study, [Listing 2](#) shows a domain ontology for a flight that agrees on the defined restrictions. It showcases the specification of ontologies in F-Logic with respect to the ontology modeling elements defined in WSMO. The ontology states the following:

- A flight has an origin and a destination, a departure and arrival date and time, and an specific price. Further, it counts with a company responsible for the flight called carrier. In particular cases, to reach a destination a flight must connect with some other, for such purpose the concept connecting flight its added. Further, every passenger receives a ticket that specifies his/her seat number, the class where is going to travel (tourist/business) and the type of ticket, single or round way.
- Location, Date, Time, Price, Carrier and Ticket are additional concepts in the ontology which are used to define necessary properties of a flight.
- Axiom ensure that the departure date is in the future, and that the connecting flight can be caught, both from a temporal and spatial point of view, and some other relevant aspects to the use case.
- Three locations are defined as instances, one for the 'Innsbruck' with the unique identifier "inn" another one for 'Galway' that has the unique identifier "gwy" and a third one for 'Dublin' with the identifier "dub". Further, a flight from Innsbruck to Dublin and its corresponding connection to Galway are also defined containing instances of ticket, price and carrier.

Listing 2 formalizes the domain knowledge.

```
To be later completed
```

Listing 2. Example for Domain Ontology Description

Ontologies are applied in every other WSMO element as domain specific vocabulary definitions. In order to support modularized development of ontologies, WSMO allows to import other ontologies into an ontology specification by the "usedMediators" – primitive. Therefore, the so called ooMediators are used to define the concrete connection between an ontology and another WSMO component (see [section 4.5](#)). In addition, ooMediator does not only connect two ontologies, but defines the possibly needed mediation facility concurrently. This concept restricts the problems of ontology integration to the mediators, thus allowing ontology development and maintenance without by interoperability problems into account. In this version of the case study, and so of the primer, the used of ooMediators is not required. Further versions of the primer will show detailed examples on how to use them.

4.3 Goals

A Goal specifies objectives that a client may have when consulting a Web Service, i.e. functionalities that a Web Service provides from the user perspective. A goal in

WSMO is described by **non-functional** properties, **post-conditions**, and **effects**. WSMO restricts the definition of goals to post-conditions and effects, leaving aside preconditions and assumptions. Post-conditions define the state of the desired information space. Effects describe the desired state of the world after the execution of the Web Service. Furthermore, a goal can import existing ontologies to make use of concepts and relations defined elsewhere, either extending or simply reusing them as appropriate, by means of **used mediators**.

4.3.1 Goal Non-functional Properties

Listing 3 provides a detailed description of the non-functional properties of the goal:

```
GNFP: [
    title ->> {Plane Itinerary Goal}
    creator ->> {Sinuhe Arroyo}
    subject ->> {travel, leisure, plane}
    description ->> {Goal that for searching, booking a
paying plane tickets}
    publisher ->> {DERI - Innsbruck}
    contributor ->> {DERI - Galway}
    date ->> {2004-03-29}
    type ->> {DataSet}
    format ->> {text plain}
    identifier ->> {gnfp-01}
    source ->> {http://www.wsmo.org/2004/d3/d3.1/v0.1}
    language ->> {English}
    relation ->> {ontology-01, webService-01}
    coverage ->> {Europe}
    rights ->> {DERI liability}
    version ->> {0.1}
]: nonFunctionalProperties
```

Listing 3. Goal Non Functional Properties

4.3.2 Post-conditions

Post-conditions define the state of the desired information space. In our concrete use case, post-conditions can be summarized as by the following statements:

- Total travel time should be smaller than 7 hours
- Traveling date is 2004-03-29
- The price limit is 600 euros
- The carrier should be any
- In the whole itinerary there should be only one plane change
- Allowed connecting destinations (if required) are Dublin, Amsterdam, Frankfurt or London
- Time between connecting flights should be smaller than 2 hours
- The ticket class is economy
- The origin is Innsbruck
- The destination is Galway

Listing 4 models the post-conditions by means of F-Logic:

```
To be later completed
```

Listing 4. Goal post-condition description

4.3.3 Effects

Currently this example only counts with one effect:

- The purchased ticket is delivered.

Description in WSMO:

```
To be later completed
```

Listing 5. Effect Description

4.3.4 Used Mediators

ggMediators can be used to describe complex goals which can be decomposed into simpler or atomic goals. In the case that similar goals are defined elsewhere, a ggMediator can then be applied to re-use such existing goal and further adopt or refine it to the target goal.

In our concrete case study the goal can be verbalized as *'buy a plane ticket to fly from Innsbruck (Austria) to Galway (Ireland)'*. Let us assume that there already exists an identified goal suitable for reuse in this concrete scenario, which allows to *'buy a product'* with product name as an attribute of the goal. In addition, there is another one which is defined as *'buy airplane ticket'*, which adds to the definition of the previous goal some other attributes such as, *'origin'*, *'destination'*, *'startDate'*, and *'endDate'*. By means of ggMediators both goals can be imported and combined to actually achieve the desire objective. In case that the terminologies used to define the two different goals are not the same, an ooMediator should be used in order to align them, as is the case in our case study. Listing 4 provides the detailed description of the ggMediator.

```
To be completed later
```

Listing 6. Goal ggMediator

4.4 Web Services

A Web Service description in WSMO consists of four sub-components:

- Non Functional Properties
- Used Mediators
- Capability
- Interfaces

Non functional properties provide relevant information for the concrete usage of a Web Service, without affecting its functional aspect. **Used Mediators** permit to import ontologies by means of ooMediators in order to reuse concepts and relations defined elsewhere. The **capability** of a Web Service defines its functionality in terms

of pre and postconditions, assumptions and effects. The **Interface** provides further information on how the functionality can be achieved, describing the communication pattern that allows to consume the functionality **choreography**, and/or how the overall functionality is achieved by means of cooperation of different service providers **orchestration**.

4.4.1 Non-functional properties

The non functional properties of Web Services in WSMO are complemented with additional ones in order to describe Quality of Service, performance aspects, security and robustness, financial information of usage, and network related issues. Following, an enumeration of non functional properties of WSMO:

```

WSNFP: [
    title ->> {Search, book and pay plane ticket}
    creator ->> {Sinuhe Arroyo}
    subject ->> {travel, leisure, plane, purchase, sell}
    description ->> {Web Service that allows to search,
book, and pay plane tickets}
    publisher ->> {DERI - Innsbruck}
    contributor ->> {DERI - Galway}
    date ->> {2004-03-29}
    type ->> {DataSet}
    format ->> {text plain}
    identifier ->> {wsnfp-01}
    source ->> {http://www.wsmo.org/2004/d3/d3.1/v0.1}
    language ->> {English}
    relation ->> {webService-01}
    coverage ->> {Europe}
    rights ->> {DERI liability}
    version ->> {0.1}
]: nonFunctionalProperties

```

Listing 7. Web Service Non Functional Properties

In addition to the core properties, WSMO allows the definition of Web Service specific non-functional properties, which allow describing quality aspects of a Web Service (QoS):

```

QoSFP: [
    performance ->> {(throughput, high), (latency, low),
(executionTime, low), (transactionTime, low), (responseTime,
low)}
    reliability ->> {(0/24)}
    security ->> {(authentication, yes),
(confidentiality, yes), (traceability, yes), (dataEncryption,
yes), (non-repudiation, yes)}
    scalability ->> {1000/1}
    robustness ->> {(incomplete, 0), (invalid, 0)}
    accuracy ->> {0/24}
    transactional ->> {Full}
    trust ->> {(high)}

```

```

    financial ->> {(variable)}
    network related QoS ->> {(delay, 0.05),
(delayVariation, +/-0.01), (messageLoss, 10/1000)}
]: QoSNonFunctionalProperties

```

Listing 8. Additional Web Service Non Functional Properties

It is important to notice that at the moment the specification does not provide a controlled vocabulary for the values of the Web Service specific non-functional properties. In future versions of the specification, recommended best practice to define this terms will be made available. In the mean while, a short description of the meaning of each value is provided:

- **Performance** is measured in terms of throuput, latency, execution time, transaction time, and response time, having each one of these parameters a valued assigned from the domain {low, medium, high}.
- **Reliability** is measured as the number of failures in a time interval. The selected time interval in this is 24 hours, which means that our service is very reliable and never fails.
- **Scalability** is measured as the number of solved requests in a time interval, in this case the time interval is 1 second.
- **Robustness** represents the ability of the service to function correctly in presence of incomplete or invalid inputs. The number provided represent the maximum allowed value for this parameters. In our concrete example the service is not able to work unless all the input parameters are complete and valid.
- **Accuracy** is measured as the number of errors in a time interval. The selected time interval is 24 hours.
- **Transactional** expresses the transactional properties of the service. The value '*full*', symbolizes that the service counts with full transactional support. Other allowed values are '*none*'.
- High **trust** indicates a elevated level of worthiness of the service. Other allowed values are '*low*' and '*medium*'.
- **Financial** expresses cost related properties of the service. Due to the nature of the service the financial information will depend upon the selected plane ticket. Allowed values for this property are either the literal '*variable*', or a concrete amount expressed in a valid currency for cost fixed services.
- In the **network related QoS** three parameters are specified:
 - **Network Delay**: Measured in milliseconds. Current network delay is 0.05 milliseconds.
 - **Delay Variation**: Measured in milliseconds. It represents the upper and lower limit of the delay. In this particular case it could go up or down 0.01 milliseconds.
 - **Message Loss**: Number of messages lost out of every 1000 sent. Currently the network does not perform really well losing 10 out of every 1000 messages.

4.4.2 Used Mediators

No mediators required.

4.4.3 Capability

The capability of a Web Service defines the service by means of its functionality. To do so it makes use of: **non functional properties, used mediators, pre-conditions, post-conditions, assumptions, and effects**. A Web Services defines one capability.

To be later completed

Listing 9. Web Service Capability

4.5 Mediators

The current version of the WSMO specification distinguishes among two types of mediators namely, **refiners** and **bridges**. While refiners allow to define a component as a refinement of an existing one, bridges provide support for re-use, by enabling two heterogeneous components to interact with each other. Currently the specification counts with four types of mediators, ggMediators and ooMediators are refiners, while wgMediators and wwMediators are counted as bridges. This general architecture allows the linking of possibly heterogeneous resources and, at the same time, include description of the mediation facility and is employed in whenever WSMO components are linked.

- [OO Mediator](#): Allow to link 2 ontologies, i.e. import of an ontology by another ontology, or use of an ontology by a component.
- [GG Mediator](#): link 2 goals, i.e. definition of sub-goal hierarchies.
- [WW Mediator](#): link 2 Web Services when they have to interact directly.
- [WG Mediator](#): link between a Web Service and a Goal, more precisely between the Capability of a Web Service and a Goal.

In this current version of the primer, a ggMediator is used to import and combine two goals, while a ooMediator is used to aligned the different ontologies used in its definition. In future versions of the primer the use of wwMediators and wgMediators will also be shown. For further reference in regard the concepts around mediators please refer to WSMO Specification: [D2v02. Web Service Modeling Ontology - Standard](#).

5. WSMO Specification Structure

The following section presents a detailed overview of the WSMO specification. It enumerates all the deliverables that form it, together with a brief description of the content of each deliverable. Further, it provides links to the latest final draft, and the latest working draft of each currently available document. At the moment of writing the primer the WSMO specification was made of 17 deliverables:

- **D1.1 - Web Site**
The WSMO web site
 - Latest Working Draft: [D1.1 Web Site](#)
- **D2 Web Service Modeling Ontology**
Presents the Web Service Modeling Ontology (WSMO) for describing various aspects related to Semantic Web Services. The work takes as starting point the Web Service Modeling Framework (WSMF).
 - Latest Final Draft: [D2v02. Web Service Modeling Ontology](#) (2004-

- 03-06)
 - Latest Working Draft: [D2v03. Web Service Modeling Ontology - Standard](#) (2004-03-08)
- **D3.1 Primer**

Provides the reader with the fundamental knowledge required to effectively use WSMO, presenting the basic concepts of the WSMO, and finally describing the content and purpose of other documents within the specification.

 - Latest Working Draft: [D3.1v01. WSMO Primer](#) (2004-04-19)
- **D3.2 Case Studies**

Describes possible use cases of Semantic Web Services and showcase how these can be modeled with WSMO.

 - Latest Working Draft: [D3.2v01. WSMO Use Case Modeling and Testing](#) (2004-04-05)
- **D4.1 Conceptual Comparison WSMO/OWL-S**

Provides a detailed comparison among the Web Service Modeling ontology (WSMO) and the Semantic Web Service Ontology (OWL-S) from a conceptual point of view. The purpose of the comparison is to highlight the conceptual overlaps and conceptual differences between the two specifications.

 - Latest Final Draft: [D4.1v01 Conceptual Comparison WSMO/OWL-S](#) (2004-03-09)
 - Latest Working Draft: [D4.1v02 Conceptual Comparison WSMO/OWL-S](#) (2004-03-09)
- **D4.2 Formal Comparison WSMO/OWL-S**

Formally compares the Web Service Modeling Ontology (WSMO) and the Semantic Web Services ontology OWL-S by hand of some well-known examples. By trying to match the chosen examples in either formalization we want to achieve a better understanding on how far one can get using either approach, which are the disjoint features, and which compatibilities exist.

 - Latest Working Draft: [D4.2v01. Formal Comparison WSMO/OWL-S](#) (2004-03-15)
- **D4.3 Mapping Tool OWL-S**

Provides the means to automatically map from OWL-S to WSMO. This shall be achieved by analysis of the components of OWL-S and translation into respective components of WSMO.
- **D5.1 Inferencing Support for the Semantic Web Services: Proof Obligations**

Presents and discusses the formal framework and tool suite that has been (resp. is in the process of being) developed along with Web Service Modeling Ontology (WSMO) and the Web Service Modeling Language (WSML) in order to provide inferencing support in the area of semantic web services.

 - Latest Working Draft: [D5.1v01. Inferencing Support for the Semantic Web](#) (2004-04-05)
- **D5.2 Inferencing Support for the Semantic Web Tools: for Semantic**

Support

Describes the approaches and tools that are applied to formally establish the identified formal proof obligations.

- Latest Working Draft: [D5.2v01. Inferencing Support for the Semantic Web Tools: for Semantic Support](#) (2004-04-05)
- **D6.1 WSMO Activity sheet**

Detailed view on the current state of deliverables and responsible persons.

 - Latest Working Draft: [D6v1. Activity Sheet](#) (2004-04-13)
- **D6.2 WSML Activity sheet**

Detailed view on the current state of deliverables and responsible persons.

 - Latest Working Draft: [D6.1v1. Activity Sheet](#) (2004-04-13)
- **D7.1 Mission Statement - WSMO**

States what the objectives and aim of WSMO are.

 - Latest Final Draft: [D7.1v1.Mission Statement - WSMO](#) (2004-01-28)
 - Latest Working Draft: [D7.1v2. Mission Statement - WSMO](#) (2004-01-28)
- **D7.2 Mission Statement - WSML**

States what the objectives and aim of WSML are.

 - Latest Final Draft: [D7.2v1.Mission Statement - WSML](#) (2004-04-01)
 - Latest Working Draft: [D7.2v1. Mission Statement - WSML](#) (2004-04-01)
- **D8 Language Evaluation and Comparison**

Investigates and analyzes several different formal and logical frameworks for describing knowledge about a static world as well as formalisms that allow to model dynamic aspects of a changing world. The deliverable aims at defining an adequate logical foundation for WSMO and for the future Web Service Modeling Language (WSML).

 - Latest Working Draft: [D8v01. Language Evaluation and Comparison](#) (2004-03-22)
- **D9 WSMO Editor**

Prototype implementation that allows to experiment with the formal description provided by WSMO.

 - Latest Working Draft: [D9v01. WSMO Editor](#) (2004-03-24)
- **D10 WSMO Registry**

Takes existing technologies like UDDI and WSIL (Web Service Inspection Language) specifies the registry for Web Services described in WSMO.

 - Latest Working Draft: [D10v01. WSMO Registry](#)(2004-03-29)
- **D11 Web Service Modeling Ontology - Lite**

Describes the WSMO - Lite ontology, a minimal subset of WSMO - Standard. WSMO-Lite follows the same structure as WSMO-Standard with the difference that WSMO-Lite simplifies by omitting specific concepts from WSMO-Standard.

- Latest Final Draft: [D11v01. Web Service Modeling Ontology - Lite](#) (2004-03-05)
- Latest Working Draft: [D11v02. Web Service Modeling Ontology - Lite](#) (2004-04-07)
- **D12 Web Service Modeling Ontology - Full**

Presents the full version of the Web Service Modeling Ontology (WSMO) for describing various aspects related to Semantic Web Services.
- **D13.0 Overview and Scope of WSMX**

The Web Services Execution Environment (WSMX) is an execution environment for dynamic discovery, selection, mediation and invocation of web services. WSMX is based on the Web Services Modelling Ontology (WSMO) which describes all aspects related to this discovery, mediation, selection and invocation.

 - Latest Working Draft: [D13.0v01. WSMX Conceptual Model](#) (2004-04-07)
- **D13.1 WSMX Conceptual Model**

Investigate and analyze several different formal and logical frameworks for describing knowledge about a static world as well as formalisms that allow to model dynamic aspects of a changing world. This first version of the deliverable focuses on WSMO-Lite.

 - Latest Working Draft: [D13.1v01. WSMX Conceptual Model](#) (2004-04-05)
- **D13.2 WSMS Execution semantics**

Defines the execution semantics of the Web Services Execution Environment.

 - Latest Working Draft: [D13.2v01. WSMX Execution Semantics](#) (2004-03-10)
- **D13.3 WSMX Mediation**

Provides an overview of the main problems that may occur during the mediation process together with the solutions that can be adopted for solving them.

 - Latest Working Draft: [D13.3v01. WSMX Mediation](#) (2004-03-29)
- **D13.4 WSMX Architecture**

Describes the WSMX architecture components.

 - Latest Working Draft: [D13.4v01. WSMX Architecture](#) (2004-04-04)
- **D13.5 WSMX Implementation**

Brings together the work described in the other deliverables of the specification. The implementation of WSMX will act as a WSMO test bed. Issues and solutions encountered in the course of the software design and development will provide a valuable source of feedback to the WSMO working group.

 - Latest Working Draft: [D13.5v01. WSMX Implementation](#) (2004-03-15)
- **D14 Choreography in WSMO**

Tackles the problem of Choreography in WSMO. It describes the necessary information for the user (human or not) of a WSMO compliant

web service to interact with it.

- Latest Working Draft: [D14v01. Choreography in WSMO](#) (2004-03-22)

- **D15 Orchestration in WSMO**

Tackles the problem of orchestration in WSMO. It describes how the service works from the provider's perspective (i.e. how a service makes use of other web service or goals in order to achieve its capability).

- **D16.0 Language Syntax for WSMO Standard**

Presents the languages used to describe the concepts defined in WSMO.

- Latest Working Draft: [D16.0v01. Language Syntax for WSMO Standard](#) (2004-04-05)

- **D16.1 Language syntaxes for WSMO Standard**

Presents the grammar for the language Web Services Modeling Language (WSML). This language is meant for representing the concepts introduced in the Web Service Modeling Ontology (WSMO).

- Latest Working Draft: [D16.1v01. Language syntaxes for WSMO Standard](#) (2004-04-05)

- **D16.2 F-logic/XML - An XML Syntax for F-logic**

Defines an XML syntax for F-logic [Kifer et al., 1995]. This syntax, henceforth referred to as F-logic/XML, captures a large and the most useful subset of F-logic, which includes rules, facts, and queries. The syntax goes far beyond Horn rules by permitting disjunction and explicit quantification.

- Latest Final Draft: [D16.2v01. F-logic/XML - An XML Syntax for F-logic](#) (2004-03-24)
- Latest Working Draft: [D16.2v01. F-logic/XML - An XML Syntax for F-logic](#) (2004-03-24)

- **D16.3 WSML/XML - An XML Syntax for WSML**

Provides an XML syntax for the Web Service Modeling Language WSML.

- Latest Working Draft: [D16.3v01. WSML/XML - An XML Syntax for WSML](#) (2004-03-22)

- **D16.4 WSML F-Logic syntax**

Provides F-Logic Syntax for the Modeling primitives outlined in the Web Service Modeling Language WSML.

- Latest Working Draft: [D16.4v0.1. WSML F-Logic Syntax](#) (2004-03-24)

- **D16.5 WSML OWL syntax**

Presents how the WSML OWL Syntax will encode that portions of WSML/WSMO that are expressible within OWL, this will ease integration for tool builders that already have OWL infrastructure in place.

- **D17 Tutorial**

The aim of this deliverable is to provide the resources to effectively disseminate the work carried within the WSMO specification.

- **D18 A Language Neutral API for Ontology Interchange**
Defines a language-neutral meta model for defining Ontologies.
 - Latest Working Draft: [D18v0.1. A Language Neutral API for Ontology Interchange](#) (2004-04-09)
- **Persistent URIs for WSMO and WSML deliverables**
Describes the way URIs (Uniform Resource Identifiers) for working drafts in the WSMO and WSML Working Groups should look.
 - Latest Working Draft: [Persistent URIs for WSMO and WSML deliverables](#) (2004-03-24)

6. References

[DCT1] DCMI Type Vocabulary. DCMI Recommendation, 11 July 2000.
<<http://dublincore.org/documents/dcmi-type-vocabulary/>>

[Gruber, 1993] Thomas R. Gruber, "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, 5:199-220, 1993.

[MIME] Internet Media Types. <<http://www.isi.edu/in-notes/iana/assignments/media-types/media-types>>

[Fensel & Bussler, 2002] D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF, Electronic Commerce Research and Applications, 1(2), 2002.

[W3CDTF] Date and Time Formats, W3C Note. <<http://www.w3.org/TR/NOTE-datetime>>

[TGN] Getty Thesaurus of Geographic Names.
<<http://www.getty.edu/research/tools/vocabulary/tgn/index.html>>

Acknowledgments

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, and SWWS; by Science Foundation Ireland under the DERI-Lion project; and by the Austrian government under the CoOperate programm.

The editors would like to thank to all the [members of the WSMO working group](#) for their advises and inputs to this document.

[webmaster](#)