



D13.1v0.1 Web Service Modeling Execution Environment - Conceptual Model (WSMX_O)

DERI Working Draft 13 July 2004

This version:

<http://www.wsmo.org/2004/d13/d13.1/v0.2/20040713/>

Latest version:

<http://www.wsmo.org/2004/d13/d13.1/v0.2/>

Previous version:

<http://www.wsmo.org/2004/d13/d13.1/v0.1/20040713/>

Authors:

Emilia Cimpian
Adrian Mocan
Matthew Moran
Eyal Oren
Michal Zaremba

Editor:

Emilia Cimpian

Reviewer:

Chris Preist

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

Table of contents

1. Introduction

2. Real-World Problem

3. Concepts

3.1. Concepts from WSMO-Lite

3.2. Concepts from WSO-Standard

3.3. Additional Concepts

3.3.1. wsmxGoal

3.3.2. Business Partner

3.3.3. Preference

3.3.4. Message

4. Conceptual model

5. Conclusions

References

Acknowledgement

1. Introduction

This document identifies and describes the concepts within a Web Service Modeling Execution Environment (WSMX) and also the relationships between these concepts. By doing this, the aim is to provide not only a common vocabulary, meant to be used at different phases of this project, but also a central point of reference for the design and development team.

To identify the WSMX concepts, a detailed description of the real-world problem we intend to solve will be provided and analyzed. The concepts are identified based strictly on this description.

The conceptual model will provide a clear and unambiguous representation of the concept's interactions. Its purpose is not to supply a description of the process flow, but to represent in real-world terms each concept's role as part of the entire system.

This document is structured in five parts. [Section 1](#) is an introduction to the topic. [Section 2](#) describes the real world problems. Sections [3](#) and [4](#) have the role of identifying the concepts and the conceptual model. The last part, [Section 5](#) contains the final conclusions.

2. Real-World Problem

For identifying the concepts involved in WSMX we start with a simple problem: two *business partners* want to communicate (for example one business partner wants to transmit a purchase order to the other). Each of them uses different *ontologies* and each of the ontologies consists of *concepts* and the *relations* between the concepts. Additionally, an ontology may use one or more *mediators*.

Each business partner can provide *Web Services*, which are described in terms of their functionalities by their *capabilities*. A Web Service may also use mediators.

Our execution environment allows a business partner to submit a request, that consists of a formal description of his goal. For making a clear differentiation between our goal and the goal defined in the Web Service Modeling Ontology Standard ([WSMO-Standard](#)), we will call this one *wsmxGoal*. The *wsmxGoal* may contain a list of *preferences* the business partner has for achieving his goal (for example, when the business partner wants to buy a certain book, he may be interested in buying it at the lowest price, from the most reliable Web Service).

There are two necessary steps for satisfying the business partner's goal: the discovery of all Web Services that may satisfy the goal and the selection of only one Web Service for the actual provision of the service. The first one deals with goal-capability matching, and the second one complying with the preferences. A possible problem that may occur during these two phases is the usage of different ontologies (the Web Services may use different ontologies from the requestor business partner). In this case, the requestor and the Web Services need to use mediators (the [WSMO- Lite ooMediator](#)), which in the first phase will provide the mapping between the goal and the capability, and in the second phase between the non functional properties of the Web Service and the preferences expressed by the

requestor.

After determining what Web Service is the best for achieving his goal, the business partner has to send a *message* to the entity that provides that Web Service.

3. Concepts

Analyzing the real-world problem description from the previous section, one can identify the following concepts: *business partner*, *ontology*, *concept* (*conceptDefinition*), *relation* (*relationDefinition*), *mediator* (we are referring here only to the *ooMediator*), *web Service*, *capability*, *wsmxGoal*, *preference* and *message*

Some of these concepts are described in Web Service Modeling Ontology-Lite ([WSMO-Lite](#)) or in [WSMO-Standard](#).

3.1. Concepts from WSMO-Lite

The concepts *ontology*, *conceptDefinition*, *relationDefinition*, *ooMediator* and *capability* are already defined in [WSMO-Lite](#), as well as the non functional properties used here, and these definitions remain unaltered.

3.2. Concepts from WSMO-Standard

The definition of a *Web Service* adopted here is the one provided in [WSMO-Standard](#). The reason we do not use the definition from [WSMO-Lite](#) is the need to refer the Web Services' non functional properties, which in [WSMO-Lite](#) are equivalent to the core properties. The preferences will refer to a subset of these non functional properties.

3.3. Additional Concepts

The concepts defined in WSMO specifications are not enough for describing our execution environment, so we need to define additional concepts. These additional concepts are not defined independently of the WSMO-Specification, most of them conform to the definitions provided in [WSMO-Lite](#) or [Standard](#), or are specializations of them.

3.3.1. wsmxgoal

The concept *wsmxgoal* inherits the *goal* defined in [WSMO-Standard](#). We chose to specialize it, and not just to adopt it, because our execution environment only supports *ooMediators*. As a consequence, the *usedMediators* attribute is overwritten to allow only *ooMediators*. The *wsmxgoal* additionally contains a list of preferences. If more than one Web Service satisfies the goal, the preferences are used for selecting only one of them.

```
wsmxgoal : : goal
wsmxgoal [
usedMediators=>>ooMediator
preferences=>preference
]
```

3.3.2. Business Partner

By the generic term of *business partner*, we understand it to mean all entities (humans or not, companies or enterprises) involved in a business process. Each of them is described by *nonFunctionalProperties* and they are able to describe their *goals*, use certain *ontologies*, and have the ability to provide a number of *web services*.

```
businessPartner [
nonFunctionalProperties =>nonFunctionalProperties
wsmxgoal =>> wsmxgoal
offeredWebServices =>> WebService
usedOntology =>> ontology
]
```

Listing 2. Business Partner definition

3.3.3. Preference

A *preference* represents a constraint on one or more of the Web Service's non functional properties. A business partner may have more than one preference, for allowing the selection of the most suited Web Service – if there is more than one service that satisfies the goal, only one is chosen based on these preferences.

A *preference* is an instance of the [WSMO-Standard axiomDefinition](#):

```
preference:axiomDefinition
```

Listing 3. Preference definition

3.3.4. Message

After determining what Web Service is the best for achieving his goal, the business partner has to send a message to the entity that provides that Web Service. A *message* is a specialization of *conceptDefinition*, adding two new elements: *targetedBusinessPartner* and *sourceBusinessPartner*. The definition of the message is:

```
message::conceptDefinition
message [
targetedBusinessPartner=>businessPartner
sourceBusinessPartner=>businessPartner
]
```

Listing 7. Message definition

4. Conceptual Model

The relationships between concepts need to be very well defined, to allow a better

understanding of the execution environment. In this chapter, a detailed description of these relationships is provided, and a graphical representation of the entire conceptual model is presented below.

The central concept of the WSMX environment is the business partner, so this will be the starting point in defining the relations across the conceptual model.

Each business partner *uses* one or more ontologies. We will consider the relationship between business partner and ontology a n to n relation: each business partner can use more than one ontology, and each ontology can be *used by* more than one partner.

Every ontology *contains* concepts and relations between concepts. There can be any number of concepts and relations in an ontology, and each one of them can be *part of* one or more ontologies. An ontology can *use* any number of ooMediators, and a mediator can be *used by* zero or more ontologies.

A business partner can *have* any number of wsmxgoals, and each wsmxgoal *specifies* a list of preferences.

For accomplishing the actual communication, a business partner can *send/receive* any number of messages. However, a message can be *sent by* only one business partner.

Figure 1 provides a graphical representation of the relationships described above:

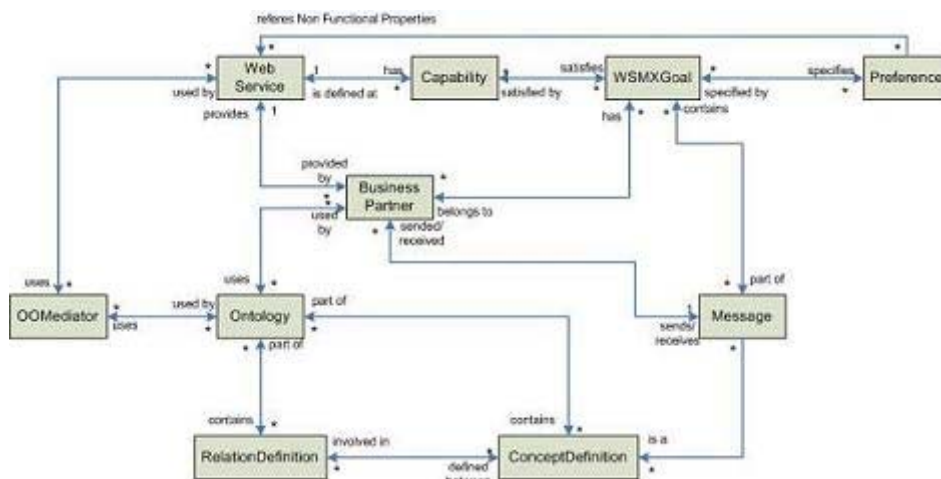


Figure 1. Conceptual Model

5. Conclusions

This document covered an important phase in the development of a software tool: the identification of the involved concepts and of the conceptual model. The purpose of developing this conceptual model is to provide a common vocabulary and a central point of reference for the design and development team.

References

[Roman et al., 2004a] D. Roman, H. Lausen, E. Oren, R. Lara (eds.): Web Service

Modeling Ontology -Lite (WSMO - Lite), version 0.1 available at
<http://www.wsmo.org/2004/d11/v0.2/>

[Roman et al., 2004b] D. Roman, H. Lausen, U. Keller (eds.): Web Service Modeling Ontology -Standard (WSMO - Standard), version 0.1 available at
<http://www.wsmo.org/2004/d2/v0.3/20040329/>

Acknowledgement

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperonto and h-TechSight; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate programme.

The editors would like to thank to all the members of the WSMO working group for their advices and inputs to this document.



[webmaster](#)