



D9.2v0.1 WSML Editor

WSMX Working Draft 31 January 2005

This version:

<http://www.wsmo.org/2005/d9/d9.2/v0.1/20050131>

Latest version:

<http://www.wsmo.org/2005/d9/d9.2/v0.1/>

Previous version:

<http://www.wsmo.org/2005/d9/d9.2/v0.1/>

Editors:

Mick Kerrigan

Authors:

Mick Kerrigan

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

Copyright © 2005 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Table of contents

1. Introduction

[1.1 Overview](#)

[1.2 Purpose of this document](#)

[1.3 Downloading the WSML Editor](#)

2. WSML Editor

[2.1 Aim](#)

[2.2 Approach](#)

[2.3 Functionality](#)

[2.3.1 Namespaces](#)

[2.3.2 Document Tree](#)

[2.3.3 Non-functional properties](#)

[2.3.4 Imported Ontologies & Used Mediators](#)

[2.3.5 Logical Expressions](#)

[2.3.6 Concepts & Relations](#)

[2.3.7 Attributes, Parameters and their values](#)

3. Future Work

[3.1 Logical Expressions](#)

[3.2 Exporting WSML](#)

[3.3 Publishing WSML](#)

[3.4 Communication with Execution Environments](#)

Appendix A: Predefined Namespaces

References

Acknowledgement

1. Introduction

1.1 Overview

The Web Service Modeling Language (WSML) [de Bruijn et al., 2004] is a formalization of the WSMO ontology [Roman et al., 2004] and provides a language for describing semantic web services. There are five languages variants, WSML-Core, WSML-DL, WSML-Flight, WSML-Rule and WSML-Full. WSML-Core, which is the intersection of Description Logic and Horn Logic, provides the basis for all the variants, while WSML-Full unites the functionality of all variants. WSML Core is extended in the direction of more expressive Description Logic by WSML-DL and in towards Logic Programming by WSML-Flight and WSML-Rule.

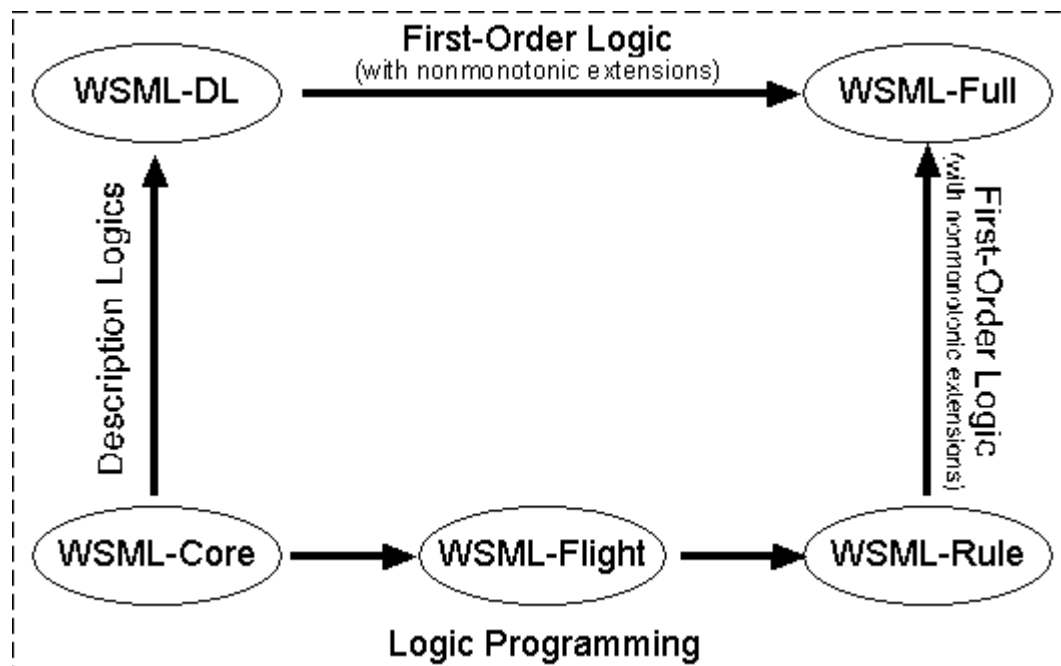


Figure 1. WSML Space [de Bruijn et al., 2004]

1.2 Purpose of this Document

The purpose of this document is to outline the WSML Editor plug-in for the Web Service Modeling Toolkit (WSMT) [Kerrigan, 2005]. The WSML Editor is a graphical tool for editing Ontologies, Goals, Mediators and Web Services described in the WSML family of languages.

1.3 Downloading the WSML Editor

The WSML Editor is distributed as a plug-in for the Web Services Modeling Toolkit (WSMT). The latest version of the WSML Editor can be obtained by downloading the latest version of the WSMT [here](#).

2. WSML Editor

2.1 Aim

The aim of developing the WSML Editor is to facilitate user experimentation with the WSMO ontology and WSML languages, as well as to provide a useful tool for describing semantic web services and publishing these descriptions to a WSMO Registry. To achieve these goals, it is necessary to hide the syntax of WSML from the users, allowing them to focus on developing descriptions of ontologies, mediators, web services, and goals without

the overhead of syntactic correctness.

2.2 Approach

The Web Service Modeling Framework (WSMF) [Fensel & Bussler, 2002] comprises four main elements that are used to describe semantic web services. As described earlier, WSML formalizes the WSMO ontology, which in turn is based on the WSMF. Therefore, the same four elements are the main elements of a WSML document. These four elements are Ontologies, Mediators, Web Services and Goals. The WSML Editor displays WSML files in a tree structure with Ontologies, Mediators, Web Services and Goals at the top level of each file. Sub elements of each type (for example Concepts, Relations, Functions, Capabilities etc.) are displayed as children of the element to which they belong.

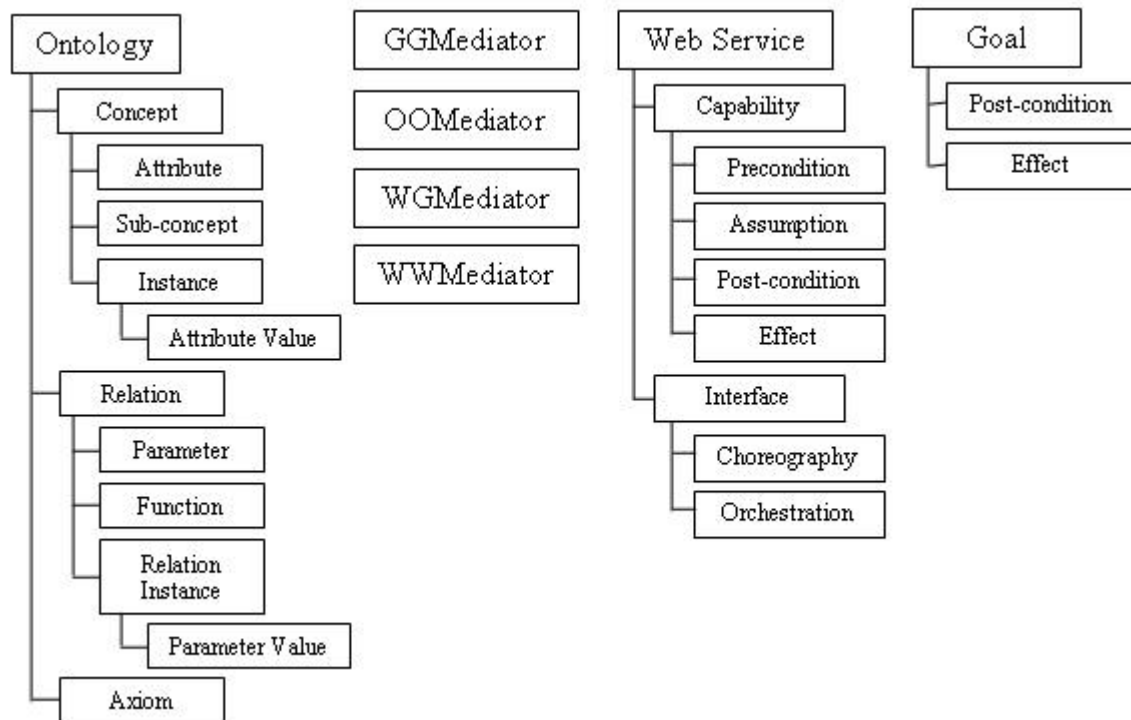


Figure 2. WSML Elements

The WSML Editor uses the libraries provided by the WSMO4J project [WSMO4J] as a basis for creating WSML documents. The WSMO4J project is an API and reference implementation compatible with WSMO v1.0 as described in WSMO Deliverable 2, Appendix A [Roman et al., 2004], and thus the WSML Editor is compatible with WSMO v1.0 also. When the format changes described in WSMO Deliverable D16.1 [de Bruijn et al., 2004] are adopted by WSMO4J, the WSML Editor will be upgraded to be compatible.

2.3 Functionality

The main aim of the first version of the WSML Editor is to support the creation of WSML files. Later work will focus on publishing descriptions to a WSMO Registry.

2.3.1 Namespaces

A WSML document contains a default namespace and a set of other namespaces, which are used to qualify names from different XML documents. This set of namespaces is made up of tuples of prefixes and URI's. All elements in a WSML document are created in the default namespace. The user can specify the default namespace at document creation time and add/remove other namespaces, while designing the elements in the WSML File. The default namespace is not be editable after document creation time, due to ongoing discussion around identifier immutability. If at any point a tuple is removed from the

namespaces which is used by an element in the WSML document, the URI will continue to be used by the element in question without a prefix.

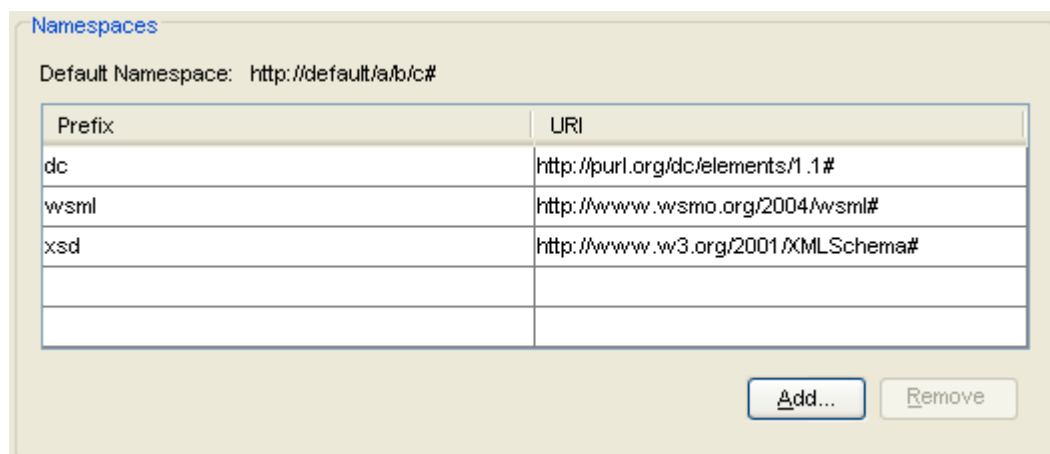


Figure 3. Namespaces

The WSML editor is deployed with a number of predefined namespaces, which the user can choose from, these namespaces include the Dublin-Core [Weibel et al. 1998], WSML and XML-Schema namespaces. By providing these namespaces to the user we are encouraging their use and preventing input errors by the user when inputting properties and values. The predefined namespaces are provided in an XML document shipped with the WSML Editor, this enables users to add namespaces to this XML document, which they use regular, to improve their own productivity. Further information on the predefined namespaces and how to locate and update the XML file shipped with the WSML Editor is available in [Appendix A](#).

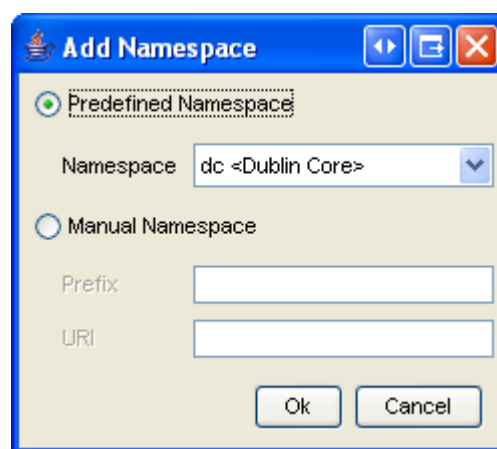


Figure 4. Add Namespace

2.3.2 Document Tree

Graphically the WSML document is broken up as laid out in Figure 2. To add a new element to the tree the user should right click on the element, which should be the super element of the new element and a list of potential new elements is displayed.

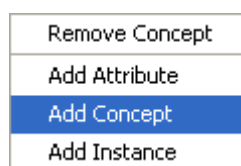


Figure 5. Document Tree Right Click

Once an element types has been selected as dialog is presented where the identifier of the new element is entered (except in the case of Attribute Values and Parameter Values, see

section 2.3.7). The identifier, which is entered must be unique for the document, except for attributes and parameters, which must be unique for the concept, relation or function they are being added to.

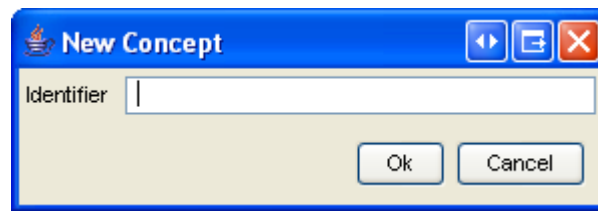


Figure 6. Add a new concept

2.3.3 Non-functional Properties

Nearly all elements in a WSML document can specified a set of non-functional properties. The recommended set of non-functional properties set by the WSML Specification [de Bruijn et al., 2004] is the Dublin-Core properties [Weibel et al. 1998], however this set is extensible by the user, for example the WSML namespace defines a number of properties not in the Dublin-Core. Both of these namespaces are included in the list of predefined namespaces shipped with the WSML Editor and more information on the non-functional properties defined by these namespaces is available in [Appendix A](#).

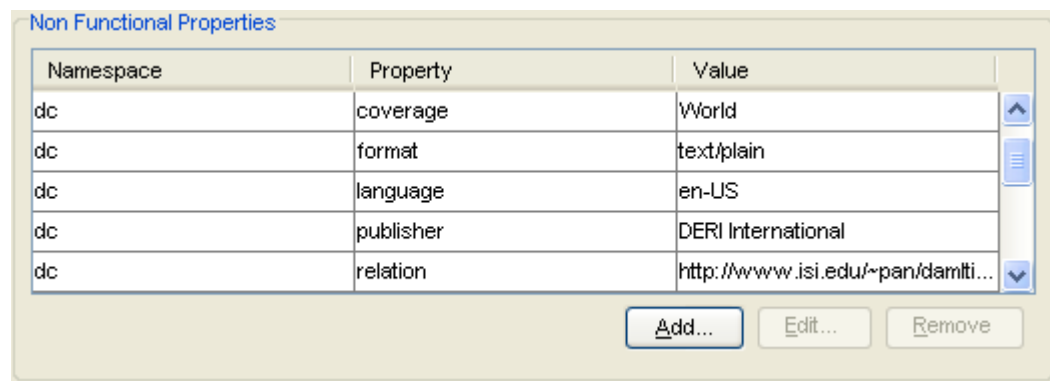


Figure 6. Non-functional Properties

As the set of non-functional properties which can be used is extensible by the user, the user can choose a predefined property or manually enter the property name and then assign a value to that property.

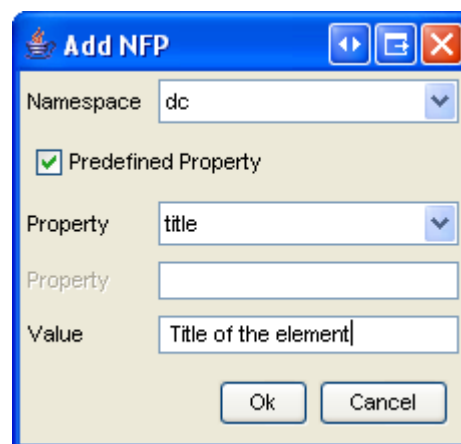


Figure 7. Add a Non-functional Property

2.3.4 Imported Ontologies & Used Mediators

Ontologies, Mediators, Web Services, Capabilities, Interfaces and Goals can all import ontologies specified in other files. Importing ontologies allows "a modular approach for

ontology design" [Roman et al., 2004]. A given ontology may be imported into multiple objects and the URI of the ontology should be included in the namespace so that it can be referenced. The mechanism for importing an ontology into an element is to first add the ontology URI to the namespace and then add the prefix to the imported ontologies of the given element by selecting the element and clicking add in the imported ontologies panel and choosing the prefix. By ensuring the user adds the URI of the ontology to the namespace, we ensure that elements in the ontology can be referenced using qualified names and that the ontology is available for importation into other elements within the WSML documents. Once imported an ontology can be removed from the list if it is no longer needed.

Ontologies, Mediators, Web Services, Capabilities, Interfaces and Goals can all use mediators to mediate between imported ontologies where necessary. Mediators are used to link elements between different ontologies to ensure heterogeneity. The user is able to add, edit and remove URI's, which identify mediators that should be used by each element to mediate between the imported ontologies and the current ontology.

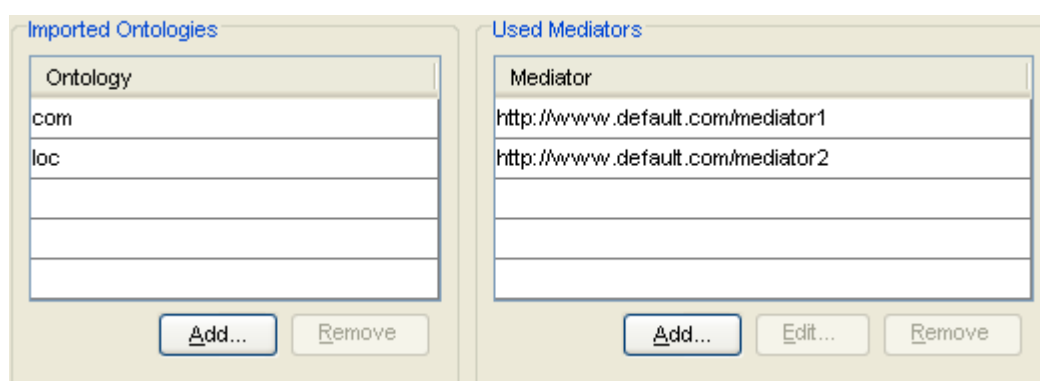


Figure 7. Imported Ontologies and Used Mediators

2.3.5 Logical Expressions

Concepts, Relations, Functions and Axioms (including capability pre-conditions, post-conditions, assumptions and effects and goal post-conditions and effects) can have a logical expression, which is used to formally specify the semantics of the concept, the set of instances of the relation or function and the statement captured by the axiom. The logical expressions are specified using the WSML Logical Expression Syntax, as defined in section 8 of the WSMO Deliverable 2 [Roman et al., 2004].

In the first version of the WSML Editor, a standard text field is provided to specify these logical expressions. Future work may include designing a graphical method for building up these logical expressions.

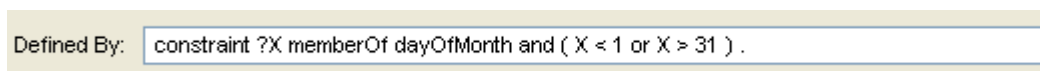


Figure 8. Logical Expressions

2.3.6 Concepts and Relations

Concepts and relations are displayed as trees in an ontology. Concepts and relations are in fact not trees but graphs, as they can have multiple super-concepts and super-relations. However performing graphing in a java applications is not straight-forward, so for the initial version of the WSML Editor a tree is used. For each concept and relation that is created within the ontology, additional super-concepts and super-relations can be added, if these super-concepts and super-relations are also specified in this ontology, i.e. not in imported ontologies, the concept is displayed at both points in the concept or relation tree. Future work may include adding graphing support to the WSML Editor.

Concept
instant

Add... Remove

Figure 9. Super Concepts

Relation
contains

Add... Remove

Figure 10. Super Relations

2.3.7 Attributes, Parameters and their values

When designing a concept a number of attributes can be created, when instantiating a concept these attributes are given values. The same is true when creating Relations and Functions except that these attributes are called parameters. An attribute and a parameter have a unique identifier (unique within the concept, relation or function where they are defined) and a type. This type can be a concept from the default namespace or any imported ontologies. An attribute has an additional boolean called 'Of Type Set', which .

Type: <Default> dayOfMonth

Of Type Set

Figure 11. Set an Attribute's Type

Type: <Default> interval

Figure 12. Set an Parameter's Type

When creating instances or relation instances it is possible to assign values to the attributes or parameters created on the super concepts, relations or functions. The user chooses one of the attributes or parameters and then assigns a value to this attribute or parameter. To aid the user in attribute and parameter identification the names are displayed as superobject.identifier, so for example if the user wished to assign a value to the 'length' attribute of the 'table' concept, then user would select 'table.length' from the list of attributes.

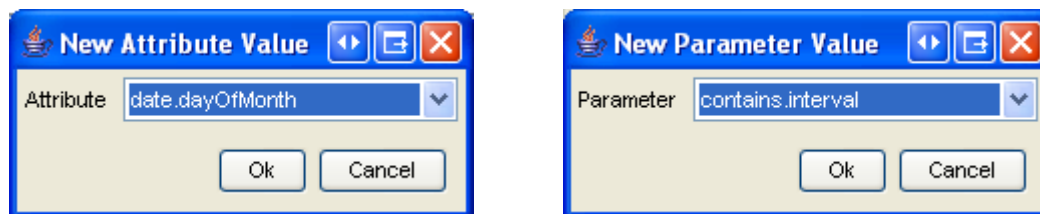


Figure 13. Add a new Attribute Value or Parameter Value



Figure 14. Set the value of an Attribute Value or Parameter Value

3. Future Work

As described in section [2.3](#), the purpose of the first version of the WSML Editor was to allow the user to create WSML documents comprising Ontologies, Mediators, Web Services and Goals. The next version of the WSML Editor will comprise updates to the current functionality and extension of the WSML Editor to allow WSML files be exported to various other formats, published to external ontology repositories and sent to execution environments (i.e. WSMX).

3.1 Logical Expressions

As described in section [2.3.5](#), logical expressions are defined in a standard text field in the current version of the WSML Editor. Part of the aim of the WSML Editor is to hide the syntax of the underlying WSML language from the user and the logical expressions are the only remaining part of the syntax in the WSML Editor. Future work will look at creating a graphical mechanism for generation of logical expressions.

3.2 Exporting WSML

It may be useful for users to convert from WSML to other file formats for modeling ontologies, for example OWL-S [[Patel-Schneider et al., 2004](#)] and FLORA-2 [[Yang et al., 2003](#)]. Future work will look at creating functionality for exporting WSML to other formats. The approach that we intend to use, is to discover semantic web services, which can perform format conversion. In certain cases it may be necessary to perform a conversion via another format i.e. WSML to OWL-S and OWL-S to FLORA-2. In doing this the WSML Editor will become a semantic web aware application and a proof of concept for semantic web service discovery, composition and invocation.

3.3 Publishing WSML

Creating a WSML document is only useful if the resulting semantic descriptions are made available for service discovery, composition and invocation. A WSML document can contain ontologies, web services, mediators and goals all of which can be published to repositories. The WSML Editor is the logical place to put functionality for interacting with these repositories. Future work will look at publishing WSML documents to and download WSML documents from these repositories.

3.4 Communication with Execution Environments

As described in section [3.3](#) creating a WSML document is only useful if the information within it can be used. Execution environments, for example WSMX [[Cimpian et al., 2005](#)],

allow the semantic descriptions to be used for service discovery, composition and invocation. The WSML Editor should allow users to communicate with these execution environments and send WSML documents to invoke features of the execution environment, for example the user should be able to realize a goal.

Appendix A: Predefined Namespaces

All files related to the WSML Editor are located within the 'wsmleditor' directory of the web services modeling toolkit installation folder. The 'namespaces.xml' file contains the list of predefined namespaces and their properties. The Dublin-Core [Weibel et al. 1998], WSML and XML-Schema namespaces are shipped by default with the WSML Editor. The following is a list of properties shipped with each of the namespaces:

Dublin-Core:

- title
- subject
- description
- type
- source
- relation
- coverage
- creator
- publisher
- contributor
- rights
- date
- format
- identifier
- language

WSML:

- version
- accuracy
- financial
- networkRelatedQoS
- performance
- reliability
- robustness
- scalability
- security
- transactional
- trust

The XML document shipped with the WSML Editor can be obtained [here](#), the file can be changed in the application folder and changes to the document will be reflected in the application once the application is restarted, the xml document should conform to the following [DTD](#).

References

[Cimpian et al., 2005] E. Cimpian, T. Vitvar, and M. Zaremba, Overview and Scope of WSMX, <http://www.wsmo.org/TR/d13/d13.0/v0.2/>

[de Bruijn et al., 2004] J. de Bruijn, H. Lausen, and D. Fensel, The WSML Family of Representation Languages, <http://www.wsmo.org/2004/d16/d16.1/v0.2/>

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

[Kerrigan, 2005] M. Kerrigan, Web Services Modeling Toolkit (WSMT), <http://www.wsmo.org/2005/d9/d9.1/v0.1/>

[Patel-Schneider et al., 2004] P. F. Patel-Schneider, P. Hayes, and I. Horrocks: OWL web ontology language semantics and abstract syntax . Recommendation 10 February 2004, W3C, 2004. Available from <http://www.w3.org/TR/owl-semantics/> .

[Roman et al., 2004] D. Roman, H. Lausen, U. Keller (eds.): Web Service Modeling Ontology (WSMO), <http://www.wsmo.org/2004/d2/v1.0/>

[Weibel et al. 1998] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf: RFC 2413 - Dublin Core Metadata for Resource Discovery

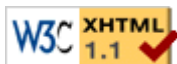
[WSMO4J] Available at <http://wsmo4j.sourceforge.net/>

[Yang et al., 2003] G. Yang, M. Kifer, and C. Zhao. FLORA-2: A rule- based knowledge representation and inference infrastructure for the semantic web. In Proceedings of the Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE) , Catania, Sicily, Italy, 2003.

Acknowledgement

This 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 WSMX working group for their advises and inputs to this document.



webmaster