



D32v0.2 WSML/RDF

WSML Working Draft 14 January 2008

This version

<http://www.wsmo.org/TR/d32/v0.2/20080114/>

Latest version

<http://www.wsmo.org/TR/d32/v0.2/>

Previous version

<http://www.wsmo.org/TR/d32/v0.1/20061220/>

Editor:

Jos de Bruijn

Authors:

Jos de Bruijn

Jacek Kopecký

Reto Krummenacher

Copyright © 2008 DERI, All Rights Reserved. DERI liability, trademark, document use, and software licensing rules apply.

Abstract

This document presents the RDF representation of WSML, called WSML/RDF. A particular WSML description is an RDF graph which uses a vocabulary defined in an RDFS ontology. WSML descriptions are generally represented as part-whole hierarchies.

All tools and resources related to WSML can be found at: <http://www.wsmo.org/wsml/wsml-syntax>

Table of Contents

- [1. Introduction](#)
- [2. RDF representation of WSML](#)
 - [2.1. General structure of WSML/RDF graph](#)
 - [2.2. IRIs and data values](#)
 - [2.3. Ontologies](#)
 - [2.4. Goals and Web Services](#)
 - [2.5. Mediators](#)
- [Appendix A. The RDFS ontology for WSML/RDF](#)
- [Appendix B. Serving WSML specifications on the Web](#)
- [References](#)
- [Acknowledgements](#)

1. Introduction

The standard representation of information on the Semantic Web is RDF. Therefore, it is advantageous to represent WSML descriptions as RDF graphs; e.g., reducing existing RDF tools, integrating various descriptions, or extending the WSML syntax in ways not foreseen by the specification.

This document specifies WSML/RDF, which is an RDFS description of the WSML vocabulary. WSML descriptions can be represented using RDF graphs which use this vocabulary. There is a straightforward correspondence between the abstract syntax of WSML [[WSML-Semantics](#)] and WSML/RDF.

2. RDF representation for WSML

Please find the complete RDF Schema for WSML in [Appendix A](#).

Each WSML description (e.g., goal, Web service, ontology, mediator) should be represented using a separate RDF graph, in a separate RDF document, in order to avoid unintended co-references. In addition, RDFS and OWL ontologies which are referenced from WSML descriptions must be represented using separate graphs as well, again avoiding unintended co-references between identifiers in the graphs.

In the remainder of the chapter, `wsml` stands for the namespace <http://www.wsmo.org/wsml/wsml-syntax#>, `rdf` stands for the namespace <http://www.w3.org/1999/02/22-rdf-syntax-ns#>, `rdfs` stands for the namespace <http://www.w3.org/2000/01/rdf-schema#>, `owl` stands for the namespace <http://www.w3.org/2002/07/owl#>, and `xsd` stands for the namespace <http://www.w3.org/2001/XMLSchema#>. Finally, `prefix:localname` stands for the concatenation of the namespace and the local name.

Annotations in WSML are represented as ordinary triples in RDF. Additionally, for each annotation *ann*, the following triple must be included in the graph (with `rdf` and `owl` denoting the RDF and OWL namespaces, respectively):

```
:ann rdf:type owl:AnnotationProperty .
```

2.1. General structure of WSML/RDF

WSML descriptions group all data related to a particular ontology, Web service, goal or mediator. In the WSML surface syntax this is achieved by grouping all descriptions under the **ontology**, **webService**, **goal**, **ggMediator**, **wgMediator**, **wwMediator**, and **ooMediator** keywords. The same holds for lower-level entities: concepts in an ontology, for example, group a number of attribute definitions and Web service capabilities group preconditions, postconditions, assumptions and effects. Conceptually, a WSML description can be seen as a part-whole hierarchy. An ontology has as parts the concept, relation, axiom, and instance definitions; in turn, these definitions are part of the ontology. Similarly for Web services, goals and mediators.

In WSML/RDF, WSML descriptions are part-whole hierarchies. For this purpose we use a part-whole ontology inspired by the work of the [Semantic Web Best Practices Working Group](#): <http://www.wsmo.org/TR/d32/v0.1/part.owl>.

Each ontology, Web service, goal, and mediator is a node in the RDF graph that is connected to all its parts using the relationship `hasPart_directly`.^[1] Figure 1 shows part of the part-whole hierarchy.

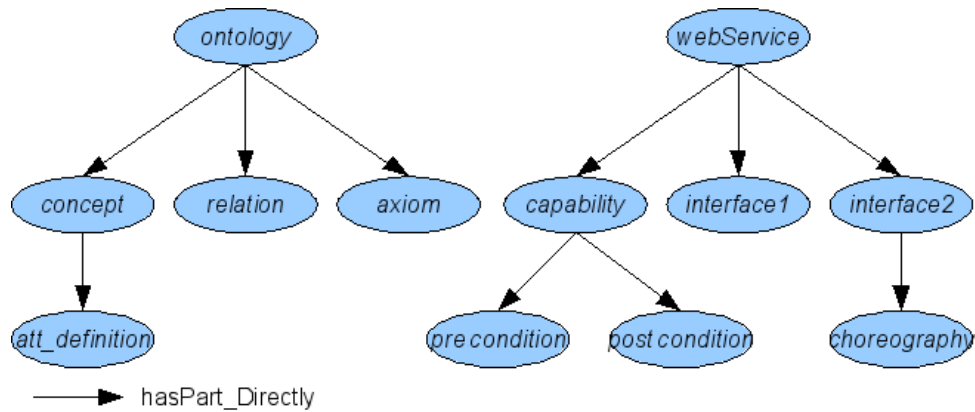


Figure 1. WSML Descriptions as part-whole hierarchy

In some cases it is necessary to disambiguate between different parts of a whole; is not possible to use the `hasPart` property for all parts of a whole. This is for example the case for the preconditions, postconditions, assumptions and effects of a capability. For example, one axiom may be a precondition of one capability and a postcondition of another capability. For this reason, we define sub-properties of `hasPart_directly`, namely `hasPrecondition`, `hasPostcondition`, `hasAssumption`, and `hasEffect`.

Logical expressions defined in axioms represented as:

- XML literals [to be defined in WSML/XML] or
- plain literals, using the WSML surface syntax, defined in [WSML], Section 2.8.

The WSML top-level entities `ontology`, `goal`, `webService`, and `mediator` correspond to RDFS classes. These RDFS classes, together with the properties which can be used in combination with these classes, are listed in Tables 1 and 2. These classes share the same superclass `wsml:TopLevelElement`.

Class	WSML entity	Has parts	Remarks
<code>wsml:TopLevelElement</code>			Superclass for all WSML top-level elements. The WSML variant is indicated using the property <code>wsml:variant</code> . Imported ontologies and used mediators are indicated using the <code>wsml:importsOntology</code> and <code>wsml:usesMediator</code> properties.
<code>wsml:Ontology</code>	<code>ontology</code> (2.3)	<code>wsml:Concept</code> , <code>wsml:Relation</code> , <code>wsml:RelationInstance</code> , <code>wsml:Axiom</code>	
<code>wsml:Goal</code>	<code>Goal</code> (2.5)	<code>wsml:Capability</code> , <code>wsml:Interface</code> , <code>wsml:NonFunctionalProperty</code>	
<code>wsml:WebService</code>	<code>WebService</code> (2.7)	<code>wsml:Capability</code> , <code>wsml:Interface</code> , <code>wsml:NonFunctionalProperty</code>	
<code>wsml:Mediator</code>	(2.6)		Source, target, and used service indicated through the properties <code>wsml:source</code> , <code>wsml:target</code> , and <code>wsml:usesService</code> , respectively.

Table 1. Classes for top-level elements of WSML/RDF

Property	WSML entity	Range	Remarks
wsml:variant	wsmlVariant (2.2.1)		Use one of the standard URIs for WSML variants.
wsml:importsOntology	importsOntology (2.2.3)	wsml:Ontology	
wsml:usesMediator	usesMediator (2.2.3)	wsml:Mediator	

Table 2. Properties for top-level elements of WSML/RDF

2.2. IRIs and data values

As can be seen from [WSML], Section 2.1.2, there are three kinds of identifiers in WSML:

- IRIs can be used directly in the RDF representation
- Data values are translated to typed XML literals in RDF, according to [WSML], Appendix C.1. For example, `_date(2005,12,2)` is translated to `"2005-12-02"^^xs:date`, assuming `xs` stands for the XML Schema datatype namespace.
- Unnumbered anonymous identifiers are represented using unique new blank nodes..

Blank nodes in RDF can be seen as existentially quantified variables, whereas anonymous identifiers in WSML can be seen as fresh constant symbols; similar to the notion of skolem constants. This apparent discrepancy between the semantics of blank nodes and the semantics of anonymous identifiers does not cause problems, because the RDF graph is merely a structural representation of the WSML description.

2.3. Ontologies

Table 3 describes the different classes used for representing entities in WSML ontologies. The section on the WSML specification [WSML] which describes the particular WSML entity is mentioned in parenthesis. The parts comprising the whole are listed in the 3rd column.

Table 4 lists the properties one can use in ontology definitions. In case there are several classes mentioned in the domain or range of a property, separated with a comma ',', these should be interpreted disjunctively, i.e., the domain (range) is a member of one of these classes, but not necessarily all. Notice that it is not possible to specify disjunctive domains and ranges in RDFS. Therefore, disjunctive domains and ranges are not explicitly specified in the RDFS ontology in Appendix A.

Class	WSML entity	Has parts	Remarks
wsml:Concept	concept (2.3.1)	wsml:attributeDefinition	is a subclass of rdfs:Class
wsml:Relation	relation (2.3.2)	rdf:List of wsml:parameterDefinition	
rdfs:Resource	instance (2.3.4)		
wsml:RelationInstance	relationInstance (2.3.4)	rdf:List of rdfs:Resource	A relationInstance has a (single) rdf:List of values, the ordered list of parameter values.
wsml:Axiom	axiom (2.3.4)		The logical expression, either as plain or XML literal, is linked to the axiom via rdfs:isDefinedBy.

wsm:AttributeDefinition	(2.3.1)		An attribute definition is part of a concept and has an associated attribute, an possibly range restrictions and min/max cardinalities.
wsm:ReflexiveAttributeDefinition	(2.3.1)		A reflexive attribute definition. This is a rdfs:subClassOf attributeDefinition.
wsm:SymmetricAttributeDefinition	(2.3.1)		A symmetric attribute definition. This is a rdfs:subClassOf attributeDefinition.
wsm:TransitiveAttributeDefinition	(2.3.1)		A transitive attribute definition. This is a rdfs:subClassOf attributeDefinition.
Table 3. Classes for ontologies in WSML/RDF			
Property	WSML entity	Domain	Range
wsm:Attribute	(2.3.1)	wsm:Concept	wsm:AttributeDefinition
wsm:hasAttributeDefinition		wsm:AttributeDefinition	wsm:AttributeDefinition
wsm:forAttribute		wsm:AttributeDefinition	wsm:AttributeDefinition
wsm:ParameterDefinition	ofType (2.3.2)	wsm:AttributeDefinition, wsm:ParameterDefinition	wsm:Concept
wsm:impliesType	impliesType (2.3.1)	wsm:AttributeDefinition, wsm:ParameterDefinition	wsm:Concept
wsm:maxCardinality	(2.3.1)	wsm:AttributeDefinition	xsd:nonNegativeInteger
wsm:minCardinality	(2.3.1)	wsm:AttributeDefinition	xsd:nonNegativeInteger
wsm:inverseOf	inverseOf (2.3.1)	wsm:AttributeDefinition	wsm:Attribute
wsm:arity	(2.3.2)	wsm:Relation	xsd:nonNegativeInteger
wsm:subRelationOf	subRelationOf (2.3.2)	wsm:Relation	wsm:Relation

Table 4. Properties for ontologies in WSML/RDF

Example 2.1 Given the following WSML ontology:

```

wsm|Variant _ "http://www.wsmo.org/wsm/wsm-syntax/wsm-flight"
namespace { _ "http://example.org/bookOntology#",
  dc _ "http://purl.org/dc/elements/1.1/",
  xsd _ "http://www.w3.org/2001/XMLSchema#"}
ontology _ "http://example.org/amazonOntology"
nonFunctionalProperties
  dc#title hasValue "Example Book ontology"
  dc#description hasValue "Example ontology about books and shopping carts"
endNonFunctionalProperties
concept book
  title ofType xsd#string

```

```

hasAuthor ofType author
concept author subConceptOf person
  authorOf inverseOf(hasAuthor) ofType book
concept cart
nonFunctionalProperties
  dc#description hasValue "A shopping cart has exactly one id
  and zero or more items, which are books."
endNonFunctionalProperties
  id ofType (1) _string
  items ofType book
instance crimeAndPunishment memberOf book
  title hasValue "Crime and Punishment"
  hasAuthor hasValue dostoyevsky

relation authorship(impliesType author, impliesType document)
nonFunctionalProperties
  dc#relation hasValue authorshipFromAuthor
endNonFunctionalProperties

axiom authorshipFromAuthor
  definedBy
    authorship(?x,?y) :- ?x[authorOf hasValue ?y] memberOf author.

```

The WSML/RDF representation is the following (using N3 notation):

```

@prefix :
  <http://example.org/bookOntology#>.
@prefix dc:
  <http://purl.org/dc/elements/1.1/>.
@prefix wsml:
  <http://www.wsmo.org/wsml/wsml-syntax#>.
@prefix part-whole:
  <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#>.
@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd:
  <http://www.w3.org/2001/XMLSchema#>.

<http://example.org/amazonOntology>
  rdf:type wsml:Ontology;
  wsml:variant, <http://www.wsmo.org/wsml/wsml-syntax/wsml-flight>
  dc:title "Example Book ontology";
  dc:description "Example ontology about books and shopping carts".
  dc:title rdf:type owl:AnnotationProperty.
  dc:description rdf:type owl:AnnotationProperty.

<http://example.org/bookOntology>
  part-whole:hasPart_directly :book;
  part-whole:hasPart_directly :author;
  part-whole:hasPart_directly :cart;

```

```
part-whole:hasPart_directly :crimeAndPunishment;  
part-whole:hasPart_directly :authorship;  
part-whole:hasPart_directly :authorshipFromAuthor.
```

```
:book  
  rdf:type wsml:Concept;  
  part-whole:hasPart_directly _:title;  
  part-whole:hasPart_directly _:hasAuthor.
```

```
_:title  
  rdf:type wsml:AttributeDefinition;  
  wsml:forAttribute :title;  
  wsml:ofType xsd:string.
```

```
_:hasAuthor  
  rdf:type wsml:AttributeDefinition;  
  wsml:forAttribute :hasAuthor;  
  wsml:ofType :author.
```

```
:author  
  rdf:type wsml:Concept;  
  rdfs:subClassOf :person;  
  part-whole:hasPart_directly _:authorOf.
```

```
_:authorOf  
  rdf:type wsml:AttributeDefinition;  
  wsml:forAttribute :authorOf;  
  wsml:ofType :book;  
  wsml:inverseOf :hasAuthor.
```

```
:cart  
  rdf:type wsml:Concept;  
  dc:description "A shopping cart has exactly one id  
    and zero or more items, which are books.";  
  part-whole:hasPart_directly _:id;  
  part-whole:hasPart_directly _:items.
```

```
_:id  
  rdf:type wsml:AttributeDefinition;  
  wsml:forAttribute :id;  
  wsml:ofType xsd:string;  
  wsml:minCardinality "1"^^xsd:nonNegativeInteger;  
  wsml:maxCardinality "1"^^xsd:nonNegativeInteger.
```

```
_:items  
  rdf:type wsml:AttributeDefinition;  
  wsml:forAttribute :items;  
  wsml:ofType :book.
```

```
:crimeAndPunishment  
  rdf:type :book;  
  :title "Crime and Punishment";  
  :hasAuthor :dostoyevsky.
```

```

:authorship
  rdf:type wsml:Relation;
  dc:relation authorshipFromAuthor;
  part-whole:hasPart_directly _:list1.

_:list1
  rdf:first _:par1;
  rdf:rest _:list2.

_:list2
  rdf:first _:par2;
  rdf:rest rdf:nil.

_:par1
  rdf:type wsml:ParameterDefinition;
  wsml:impliesType :author.

_:par2
  rdf:type wsml:ParameterDefinition;
  wsml:impliesType :document.

:authorshipFromAuthor
  rdf:type wsml:Axiom;
  rdfs:isDefinedBy "<impliedByLP><atom name="http://example.org/bookOntology#
authorship">
  <term name="?x"/><term name="?y"/></atom>
  <and>
    <molecule><term name="?x"/><isa type="memberOf"><term name="?y"/></isa></
molecule>
    <molecule><term name="?x"/>
      <attributeValue>
        <term name="http://example.org/bookOntology#authorOf"/>
        <term name="?y"/>
      </attributeValue>
    </molecule>
  </and></impliedByLP>"^^rdf:XMLLiteral.

```

2.4. Goals and Web Services

Tables [5](#) and [6](#), respectively, describe the classes and properties used for goals and web services in WSML/RDF.

Class	WSML entity	Has parts	Remarks
wsml:Capability	capability (2.4.1)	wsml:Axiom, wsml:NonFunctionalProperty	The parts of the capability, namely the axioms, are subdivided into preconditions, postconditions, assumptions and effects by the use of the properties wsml:hasPrecondition, wsml:hasPostcondition, wsml:hasAssumption, and wsml:hasEffect, respectively. Each of the shared variables is one of the values of the wsml:sharedVariable property. Variables are alphanumeric strings preceded with a question mark '?'.
wsml:Interface	interface (2.4.2)	wsml:Choreography, wsml:Orchestration, wsml:NonFunctionalProperty	
wsml:NonFunctionalProperty	nonFunctionalProperty (2.4.3)		A nonfunctional property has exactly one of a value or a variable, and a number of logical expressions, which are referred to using the properties wsml:hasValue, wsml:sharedVariable, and rdfs:isDefinedBy.
wsml:Orchestration	orchestration (2.4.2)		Orchestrations are external to WSML.
wsml:Choreography	choreography (2.4.2)	wsml:StateSignature, wsml:TransitionRule	Annotations, imported ontologies, and used mediators are referenced in the usual way.
wsml:StateSignature	stateSignature (2.4.2)	wsml:Mode	Annotations, imported ontologies, and used mediators are referenced in the usual way.

wsml:Mode	(2.4.2)	wsml:Mode	Mode of a concept or relation in the state signature of a choreography. Type of a mode is indicated through membership of exactly one of wsml:StaticMode, wsml:InMode, wsml:OutMode, wsml:SharedMode, and wsml:ControlledMode. The concept or relation to which the mode refers is indicated through the wsml:forConcept or wsml:forRelation property, respectively. The groundings to which the mode refers are indicated through the wsml:hasGrounding property.
wsml:StaticMode	static (2.4.2)		
wsml:InMode	in (2.4.2)		
wsml:OutMode	out (2.4.2)		
wsml:SharedMode	shared (2.4.2)		
wsml:ControlledMode	controlled (2.4.2)		
wsml:TransitionRule	transitionRules (2.4.2)	wsml:TransitionRule, variables (alphanumeric strings preceded with "?"), logical expressions	The type of a transition rule is indicated through membership of exactly one of wsml:IfRule, wsml:ForallRule, wsml:ChooseRule, wsml:PipedRule, wsml:AddRule, or wsml>DeleteRule. Depending on the type, a transition rule (whole) contains a logical expression and a number of rules (if rule), a number of variables, a logical expression and a number of rules (forall and choose rule), a number of rules (piped rule), or a logical expression which represents a WSML fact (add or delete rule) (parts).
Table 5. Classes for goals and web services in WSML/RDF			
Property	WSML entity	Domain	Range
wsml:hasPrecondition	precondition (2.4.1)	logical expressions, wsml:Capability	wsml:Axiom

Table 6. Properties for goal and web services in WSML/RDF

wsml:hasPostcondition	postcondition (2.4.1)	wsml:Capability	wsml:Axiom
wsml:hasAssumption	assumption (2.4.1)	wsml:Capability	wsml:Axiom
wsml:hasEffect	effect (2.4.1)	wsml:Capability	wsml:Axiom
wsml:sharedVariable	sharedVariables (2.4.1)		xsd:string
wsml:GGMediator	ggMediator (2.6.3)		
wsml:WWMediator	wwMediator (2.6.3)		Sub-class of wsml:Mediator
wsml:GGMediator	ggMediator (2.6.3)		Sub-class of wsml:Mediator
wsml:WGMediator	wgMediator (2.6.4)		Sub-class of wsml:Mediator

Table 7. Classes for mediators in WSML/RDF

Property	WSML entity	Domain	Range
wsml:source	source (2.6)	wsml:Mediator	wsml:Mediator, wsml:Ontology, wsml:Goal, wsml:WebService
wsml:target	source (2.6)	wsml:Mediator	wsml:Mediator, wsml:Ontology, wsml:Goal, wsml:WebService
wsml:usesService	usesService (2.6)	wsml:Mediator	wsml:Mediator, wsml:Goal, wsml:WebService

Table 8. Properties for mediators in WSML/RDF

Appendix A. The RDFS ontology for WSML/RDF

The complete RDF Schema for WSML can also be found [here](#).

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY wsml 'http://www.wsmo.org/wsml/wsml-syntax#'>
  <!ENTITY wsmo 'http://www.wsmo.org/TR/d2/#'>
  <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
  <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
  <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
  <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
  <!ENTITY part-whole 'http://www.wsmo.org/TR/d32/v0.1/part.owl#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
  xmlns:wsml="&wsml;"
  xmlns:rdfs="&rdfs;"
  xmlns:wsmo="&wsmo;"
  xmlns:dc="&dc;"
  xmlns:owl="&owl;"
  xmlns:part-whole="&part-whole;"
  xml:base="http://www.wsmo.org/wsml/wsml-syntax">

<owl:Ontology rdf:about="">
```

```
<rdfs:isDefinedBy rdf:resource="http://www.wsmo.org/TR/d16/d16.1/v0.3/" />
<rdfs:isDefinedBy rdf:resource="http://www.wsmo.org/TR/d16/d16.3/v0.3/" />
<rdfs:comment>
```

This ontology is a vocabulary definition for the RDF representation of WSML descriptions, i.e., WSML ontologies, goals, Web services, and mediators. Several such descriptions may occur in one graph.

However, one must be aware of the co-reference between the definitions in the various descriptions; there

is no such co-reference in the surface syntax of WSML.

Note that WSML descriptions may import RDFS or OWL ontologies. Such RDFS or OWL ontologies must be represented using separate RDF graphs to avoid co-reference issues.

In the definition of the WSML vocabulary, we reuse the following vocabularies:

```
RDF: http://www.w3.org/1999/02/22-rdf-syntax-ns#
RDFS: http://www.w3.org/2000/01/rdf-schema#
XSD: http://www.w3.org/2001/XMLSchema#
OWL: http://www.w3.org/2002/07/owl#
DC (Dublin Core): http://purl.org/dc/elements/1.1/
Part-Whole relations (from the SWBP WG): http://www.wsmo.org/TR/d32/v0.1/part.owl#
see also: http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/
```

Annotations are captured using owl:AnnotationProperty. Each annotation attribute has to be of rdf:type owl:AnnotationProperty

Note that this ontology was written in the spirit of RDF: any RDF processor may use the statements which it can work with and will ignore all statements it cannot work with. For example, in case the RDF processor cannot deal with owl:sameAs, it will not. This ontology was explicitly not constructed to fall inside any of the species of OWL, although it necessarily falls inside OWL Full, since every RDF graph is a valid OWL Full ontology.

```
</rdfs:comment>
<owl:versionInfo>$Date: 2008-01-14 17:03:10 $</owl:versionInfo>
</owl:Ontology>
```

```
<!-- WSML top-level entities -->
```

```
<rdfs:Class rdf:ID="TopLevelElement"
  rdfs:label="WSML Top-Level Element">
  <rdfs:comment>
```

Top level elements are those things which have a variant associated with them.

The top level elements are ontologies, goals, Web services, mediators, interfaces, and capabilities.

Imported ontologies and used mediators are indicated using the wsml:importsOntology and wsml:usesMediator properties.

```
</rdfs:comment>
<dc:relation rdf:resource="&wsmo;wsmoTopLevelElement"/>
```

```
<rdfs:subClassOf rdf:resource="#TopLevelElement"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="Ontology"
  rdfs:label="Ontology">
```

```
<rdfs:comment>
```

An ontology can be seen as a part-whole hierarchy. An ontology (whole) has parts concepts, relations, axioms and relation instances. A concept (whole) has parts attribute definitions.

Note that an ontology may also have annotations. Annotations are RDF triples, where the subject is the identifier of the resource to be annotated (e.g. an ontology), the predicate is the annotation property, and the object is the value. Additionally, the annotation property has to be declared to be of `rdf:type owl:AnnotationProperty`.

It is related to `owl:ontology`, but not the same, nor equivalent, because there are many ontologies which are `wsm` ontologies, but not `OWL` ontologies (e.g. `nonmonotonic` ontologies).

This class is the class of `WSML` ontologies, i.e., ontologies written using the `WSML` ontology language. It is also possible to use `RDFS` and `OWL` ontologies in `WSML` Web service descriptions. However, such ontologies must reside in separate graphs to avoid ambiguity.

Note that, although some `RDFS` vocabulary is used in `WSML` ontologies and `WSML` vocabulary is

related to the `RDFS` and `OWL` vocabularies, the semantics of `WSML` ontologies is not defined as

an extension of either `RDFS` or `OWL`. Therefore, the `RDF` representation of `WSML` ontologies is

merely a structural description. For a description of the semantics of `WSML` ontologies please refer to <http://www.wsmo.org/TR/d16/d16.3/v0.3/>.

```
</rdfs:comment>
```

```
<dc:relation rdf:resource="&owl;Ontology"/>
```

```
<dc:relation rdf:resource="&wsmo;ontology"/>
```

```
<rdfs:subClassOf rdf:resource="#TopLevelElement"/>
```

```
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="WebService"
  rdfs:label="WebService">
```

```
<rdfs:comment>
```

A web service can be seen as a part-whole hierarchy. A web service (whole) has parts (at most

one) capability and (possibly multiple) interfaces and nonfunctional properties.

```
</rdfs:comment>
```

```
<dc:relation rdf:resource="&wsmo;webService"/>
```

```
<rdfs:subClassOf rdf:resource="#TopLevelElement"/>
```

```
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="Goal"
  rdfs:label="Goal">
```

```
<rdfs:comment>
```

A goal can be seen as a part-whole hierarchy. A goal (whole) has parts (at most one) capability

and (possibly multiple) interfaces and nonfunctional properties.

```
</rdfs:comment>
<dc:relation rdf:resource="#wsmo:goal"/>
<rdfs:subClassOf rdf:resource="#TopLevelElement"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="Mediator"
  rdfs:label="Mediator">
```

```
<rdfs:comment>
  A mediator can be seen as a part-whole hierarchy. A mediator (whole) has parts (possibly
  multiple)
  nonfunctional properties.
```

A mediator may have a source and multiple targets and may use a service for its implementation;

these are indicated through the properties `wsmo:source`, `wsmo:target`, and `wsmo:usesService`, respectively. Annotations, imported ontologies, and used mediators are referenced in the usual way.

```
</rdfs:comment>
<dc:relation rdf:resource="#wsmo:mediator"/>
<rdfs:subClassOf rdf:resource="#TopLevelElement"/>
</rdfs:Class>
```

```
<rdf:Property rdf:ID="importsOntology"
  rdfs:label="importsOntology">
```

```
<rdfs:comment>
  Any WSML entity (goal, webService, ontology, mediator, interface, capability,
  choreography) may import WSML, RDFS, or OWL ontologies in order to reuse
  vocabularies.
```

This property is a super property of the `owl:imports` property.

```
</rdfs:comment>
</rdf:Property>
```

```
<rdf:Property rdf:ID="variant"
  rdfs:label="variant">
```

```
<rdfs:comment>
  A WSML entity (goal, webService, mediator, ontology) may have a WSML variant
  associated with it.
```

Defined values:

```
http://www.wsmo.org/wsml/wsml-syntax/wsml-full
http://www.wsmo.org/wsml/wsml-syntax/wsml-rule
http://www.wsmo.org/wsml/wsml-syntax/wsml-flight
http://www.wsmo.org/wsml/wsml-syntax/wsml-dl
http://www.wsmo.org/wsml/wsml-syntax/wsml-core
```

In case the same ontology, goal, etc. has different variants associated with it, the highest variant is chosen.

```
</rdfs:comment>
```

```

    <rdfs:domain rdf:resource="#TopLevelElement"/>
  </rdf:Property>

  <rdf:Property rdf:ID="usesMediator"
    rdfs:label="usesMediator">
    <rdfs:comment>Any WSML entity may use a mediator.</rdfs:comment>
    <rdfs:range rdf:resource="#Mediator"/>
  </rdf:Property>

  <!-- Ontologies -->

  <rdfs:Class rdf:ID="Attribute"
    rdfs:label="Attribute">
    <rdfs:comment>An attribute is a specific type of rdf:Property.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#rdf:Property"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="AttributeDefinition"
    rdfs:label="AttributeDefinition">
    <rdfs:comment>
      A concept may have a number of attribute definitions. An attribute
      definition consists of an attribute and possible inverse attribute,
      and maximal and minimal cardinality definitions.
    </rdfs:comment>
    <dc:relation>&owl;Restriction</dc:relation>
  </rdfs:Class>

  <rdfs:Class rdf:ID="ReflexiveAttributeDefinition"
    rdfs:label="ReflexiveAttributeDefinition">
    <rdfs:comment>Reflexive attribute definition</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#AttributeDefinition"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="SymmetricAttributeDefinition"
    rdfs:label="SymmetricAttributeDefinition">
    <rdfs:comment>Symmetric attribute definition</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#AttributeDefinition"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="TransitiveAttributeDefinition"
    rdfs:label="TransitiveAttributeDefinition">
    <rdfs:comment>Transitive attribute definition</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#AttributeDefinition"/>
  </rdfs:Class>

  <rdf:Property rdf:ID="hasAttributeDefinition"
    rdfs:label="hasAttributeDefinition">
    <rdfs:comment>A concept has zero or more attribute definitions.</rdfs:comment>
    <rdfs:range rdf:resource="#AttributeDefinition"/>
    <rdfs:domain rdf:resource="#Concept"/>
    <rdfs:subPropertyOf rdf:resource="#part-whole;hasPart_directly"/>
    <rdfs:subPropertyOf rdf:resource="#superClassOf"/>
  </rdf:Property>

```

```
<rdf:Property rdf:ID="superClassOf"
  rdfs:label="superClassOf">
  <rdfs:comment>Inverse of the rdf subclass relation.</rdfs:comment>
  <owl:inverseOf rdf:resource="#&rdfs;subClassOf"/>
</rdf:Property>

<rdf:Property rdf:ID="forAttribute"
  rdfs:label="forAttribute">
  <rdfs:comment>An attribute definition is associated with one attribute.</rdfs:comment>
  <rdfs:range rdf:resource="#Attribute"/>
  <rdfs:domain rdf:resource="#AttributeDefinition"/>
  <owl:equivalentProperty rdf:resource="#&owl;onProperty"/>
</rdf:Property>

<rdf:Property rdf:ID="ofType"
  rdfs:label="ofType">
  <rdfs:comment>
    The attribute value or relation parameter definition is checked to
    be of the specific type.
  </rdfs:comment>
  <rdfs:range rdf:resource="#Concept"/>
</rdf:Property>

<rdf:Property rdf:ID="impliesType"
  rdfs:label="impliesType">
  <rdfs:comment>
    Attribute values and relation parameter definitions are inferred to have a
    particular type.
  </rdfs:comment>
  <rdfs:range rdf:resource="#Concept"/>
  <owl:equivalentProperty rdf:resource="#&owl;allValuesFrom"/>
</rdf:Property>

<rdf:Property rdf:ID="maxCardinality"
  rdfs:label="maxCardinality">
  <rdfs:comment>
    An attribute definition may have a maximal cardinality. If no maximal cardinality is
    described, there is not constraint on the maximal cardinality.
  </rdfs:comment>
  <rdfs:range rdf:resource="#&xsd;nonNegativeInteger"/>
  <rdfs:domain rdf:resource="#AttributeDefinition"/>
</rdf:Property>

<rdf:Property rdf:ID="minCardinality"
  rdfs:label="minCardinality">
  <rdfs:comment>
    An attribute definition may have a minimal cardinality. If no minimal cardinality
    is described, the attribute is optional.
  </rdfs:comment>
  <rdfs:range rdf:resource="#&xsd;nonNegativeInteger"/>
```

```

    <rdfs:domain rdf:resource="#AttributeDefinition"/>
  </rdf:Property>

```

```

<rdf:Property rdf:ID="inverseOf"
  rdfs:label="inverseOf">
  <rdfs:comment>
    An attribute definition may have an inverse attribute
    associated with it.
  </rdfs:comment>
  <rdfs:range rdf:resource="#Attribute"/>
  <rdfs:domain rdf:resource="#AttributeDefinition"/>
</rdf:Property>

```

```

<rdfs:Class rdf:ID="Axiom"
  rdfs:label="Axiom">
  <rdfs:comment>
    An axiom is an arbitrary logical specification which can be used
    as part of WSML ontology, but also for the specification of the
    functionality of web services and goals through capabilities.

```

The logical expression itself is either expressed using the WSML/XML syntax for logical expressions as an XML literal, or using the WSML logical expression syntax, as a plain literal. In either case, the logical expression is linked to the axiom using `rdfs:isDefinedBy`.

```

  </rdfs:comment>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="Concept"
  rdfs:label="Concept">
  <rdfs:comment>
    A type of rdfs:Class.
    A concept may have a number of attribute definitions associated with
    it through the hasAttributeDefinition relationship.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#&rdfs;Class"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="ParameterDefinition"
  rdfs:label="ParameterDefinition">
  <rdfs:comment>
    A parameter definition consists of a type (ofType/impliesType)
    for the parameter, either via the property impliesType or ofType. A relation
    contains a single rdf:list of parameter definitions.
  </rdfs:comment>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="Relation"
  rdfs:label="Relation">
  <rdfs:comment>
    Super-relations are indicated through the wsml:subRelationOf property. The
    property wsml:hasParameterList is used to associate a list of
    wsml:ParameterDefinitions with the relation. Note that multiple parameter

```

definitions correspond to multiple definitions, where each definition corresponds to a relation with the same name but possibly a different arity. Alternatively, a relation may have an arity associated with it using the property `wsml:hasArity`. Note that a relation must have an associated arity or parameter list.

An RDF property can be seen as a specific kind of WSML relation, namely a binary relation.

```
</rdfs:comment>
<dc:relation>&rdf;Property</dc:relation>
</rdfs:Class>
```

```
<rdf:Property rdf:ID="hasArity"
  rdfs:label="hasArity">
  <rdfs:comment>
    A relation has either an explicit arity or an explicit parameter
    definition, which is an rdf:list.
  </rdfs:comment>
  <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
  <rdfs:domain rdf:resource="#Relation"/>
</rdf:Property>
```

```
<rdf:Property rdf:ID="subRelationOf"
  rdfs:label="subRelationOf">
  <rdfs:comment>
    A relation may be a subrelation of a number of other relations.
    A subrelation must have the same arity as the super relation.
  </rdfs:comment>
  <rdfs:range rdf:resource="#Relation"/>
  <rdfs:domain rdf:resource="#Relation"/>
</rdf:Property>
```

```
<rdfs:Class rdf:ID="RelationInstance"
  rdfs:label="RelationInstance">
  <rdfs:comment>
    A relation instance is an actual ground fact which corresponds
    to a particular relation; it can be seen as a ground atomic formula
    in predicate calculus.
```

A relation instance (whole) `hasPart` an `rdf:list` of parameter values.

The relation of which a particular `wsml:RelationInstance` is an instance is indicated using the `rdf:type` property.

```
</rdfs:comment>
</rdfs:Class>
```

```
<!-- Capabilities -->
```

```
<rdfs:Class rdf:ID="Capability"
  rdfs:label="Capability">
  <rdfs:comment>
    A capability has a number of postconditions, preconditions, effects, and assumptions, as
    well as a list
```

of shared variables, which are referred to using the properties `wsml:hasPrecondition`, `wsml:haspostcondition`, `wsml:hasAssumption`, and `wsml:sharedVariables`.

A capability (whole) hasPart a a number of nonfunctional properties.

```
</rdfs:comment>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="NonFunctionalProperty"
  rdfs:label="Nonfunctional property">
  <rdfs:comment>
```

A nonfunctional property has exactly one of a value or a variable, and a number of logical expressions,

which are referred to using the properties `wsml:hasValue`, `wsml:sharedVariable`, and `rdfs:isDefinedBy`.

```
</rdfs:comment>
</rdfs:Class>
```

```
<rdf:Property rdf:ID="hasValue"
  rdfs:label="hasValue">
  <rdfs:domain rdf:resource="#NonFunctionalProperty"/>
</rdf:Property>
```

```
<rdf:Property rdf:ID="hasPrecondition"
  rdfs:label="hasPrecondition">
  <rdfs:comment>
    A pre-condition is an axiom which expresses conditions over the input of the service.
  </rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&part-whole;hasPart_directly"/>
  <rdfs:domain rdf:resource="#Capability"/>
  <rdfs:range rdf:resource="#Axiom"/>
</rdf:Property>
```

```
<rdf:Property rdf:ID="hasPostcondition"
  rdfs:label="hasPostcondition">
  <rdfs:comment>
    A post-condition is an axiom which describes the relation between the input and the
    output of the service, as well as conditions which are guaranteed to hold over the output.
  </rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&part-whole;hasPart_directly"/>
  <rdfs:domain rdf:resource="#Capability"/>
  <rdfs:range rdf:resource="#Axiom"/>
</rdf:Property>
```

```
<rdf:Property rdf:ID="hasAssumption"
  rdfs:label="hasAssumption">
  <rdfs:comment>
    An assumption is an axiom which describes conditions on the state of the world which
    must hold for the web service to be able to execute.
  </rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&part-whole;hasPart_directly"/>
  <rdfs:domain rdf:resource="#Capability"/>
  <rdfs:range rdf:resource="#Axiom"/>
</rdf:Property>
```

```

<rdf:Property rdf:ID="hasEffect"
  rdfs:label="hasEffect">
  <rdfs:comment>
    An effect is an axiom which describes conditions which are guaranteed to hold over the
    state of the world after execution of the service.
  </rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="&part-whole;hasPart_directly"/>
  <rdfs:domain rdf:resource="#Capability"/>
  <rdfs:range rdf:resource="#Axiom"/>
</rdf:Property>

```

```

<rdf:Property rdf:ID="sharedVariable"
  rdfs:label="sharedVariable">
  <rdfs:comment>
    A capability or nonfunctional property has a number of variables which are shared
    across all definitions.
    By default, all free variables in a logical expression are implicitly universally
    quantified over this logical expression. Shared variables are free in the logical
    expression and a universally quantified over the entire capability.

    Each of the shared variables is one of the values of the wsml:sharedVariable property.
    Variables are alphanumeric strings preceded with a question mark '?'.
  </rdfs:comment>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>

```

```

<!-- Interfaces -->

```

```

<rdfs:Class rdf:ID="Interface"
  rdfs:label="Interface">
  <rdfs:comment>
    An interface (whole) may have a number of nonfunctional properties, choreographies and
    orchestrations (parts) associated with it. Nonfunctional properties, annotations,
    imported ontologies, and used mediators are associated with interfaces in the usual way.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#TopLevelElement"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="Orchestration"
  rdfs:label="Orchestration">
  <rdfs:comment>
    A Web Service interface may have an orchestration associated with it.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#TopLevelElement"/>
</rdfs:Class>

```

```

<!-- Choreographies -->

```

```

<rdfs:Class rdf:ID="Choreography"
  rdfs:label="Choreography">
  <rdfs:comment>
    A choreography (whole) contains one state signature and a number of rules (parts).

```

```

    Annotations, imported ontologies, and used mediators are referenced in the usual way.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#TopLevelElement"/>
  <dc:relation rdf:resource="&wsmo;choreography"/>
</rdfs:Class>

<rdfs:Class rdf:ID="StateSignature"
  rdfs:label="State signature">
  <rdfs:comment>
    A state signature (whole) contains a number of modes (parts).
    Annotations, imported ontologies, and used mediators are referenced in the usual way.
  </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="Mode"
  rdfs:label="Mode">
  <rdfs:comment>
    Mode of a concept or relation in the state signature of a choreography. Type of a
    mode is indicated through membership of exactly one of wsml:StaticMode, wsml:InMode,
    wsml:OutMode, wsml:SharedMode, and wsml:ControlledMode. The concept or relation to
    which the mode refers is indicated through the wsml:forConcept or wsml:forRelation
    property, respectively. The groundings to which the mode refers are indicated
    through the wsml:hasGrounding property.

    For information about types of modes and grounding see:
    http://www.wsmo.org/TR/d14/v1.0/#chorSig
  </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="StaticMode"
  rdfs:label="Static Mode">
  <rdfs:subClassOf rdf:resource="#Mode"/>
</rdfs:Class>

<rdfs:Class rdf:ID="InMode"
  rdfs:label="In Mode">
  <rdfs:subClassOf rdf:resource="#Mode"/>
</rdfs:Class>

<rdfs:Class rdf:ID="OutMode"
  rdfs:label="Out Mode">
  <rdfs:subClassOf rdf:resource="#Mode"/>
</rdfs:Class>

<rdfs:Class rdf:ID="SharedMode"
  rdfs:label="Shared Mode">
  <rdfs:subClassOf rdf:resource="#Mode"/>
</rdfs:Class>

<rdfs:Class rdf:ID="ControlledMode"
  rdfs:label="Controlled Mode">
  <rdfs:subClassOf rdf:resource="#Mode"/>
</rdfs:Class>

```

```

<rdf:Property rdf:ID="forConcept"
  rdfs:label="forConcept">
  <rdfs:comment>
    The concept of a mode must be either a concept in a WSML ontology, a class in an
    RDFS ontology, or class in an OWL ontology.
  </rdfs:comment>
  <rdfs:domain rdf:resource="#Mode"/>
</rdf:Property>

<rdf:Property rdf:ID="forRelation"
  rdfs:label="forRelation">
  <rdfs:comment>
    The relation of a mode must be either a relation in a WSML ontology, a property in an
    RDFS ontology, or an object or datatype property in an OWL ontology.
  </rdfs:comment>
  <rdfs:domain rdf:resource="#Mode"/>
</rdf:Property>

<rdf:Property rdf:ID="hasGrounding"
  rdfs:label="hasGrounding">
  <rdfs:comment>
    The relation of a mode must be either a relation in a WSML ontology, a property in an
    RDFS ontology, or an object or datatype property in an OWL ontology.
  </rdfs:comment>
  <rdfs:domain rdf:resource="#Mode"/>
  <rdfs:range rdf:resource="#Grounding"/>
</rdf:Property>

<rdfs:Class rdf:ID="TransitionRule"
  rdfs:label="Transition rule">
  <rdfs:comment>
    The type of a transition rule is indicated through membership of exactly one of
    wsml:IfRule, wsml:ForallRule, wsml:ChooseRule, wsml:PipedRule, wsml:AddRule, or
    wsml>DeleteRule. Depending on the type, a transition rule (whole) contains a logical
    expression and a number of rules (if rule), a number of variables, a logical expression
    and a number of rules (forall and choose rule), a number of rules (piped rule), or a
    logical expression which represents a WSML fact (add or delete rule) (parts).
  </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="IfRule"
  rdfs:label="If rule">
  <rdfs:subClassOf rdf:resource="#TransitionRule"/>
</rdfs:Class>

<rdfs:Class rdf:ID="ForallRule"
  rdfs:label="Forall rule">
  <rdfs:subClassOf rdf:resource="#TransitionRule"/>
</rdfs:Class>

<rdfs:Class rdf:ID="ChooseRule"
  rdfs:label="Choose rule">

```

```

    <rdfs:subClassOf rdf:resource="#TransitionRule"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="PipedRule"
  rdfs:label="Piped rule">
  <rdfs:subClassOf rdf:resource="#TransitionRule"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="AddRule"
  rdfs:label="Add rule">
  <rdfs:subClassOf rdf:resource="#TransitionRule"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="DeleteRule"
  rdfs:label="Delete rule">
  <rdfs:subClassOf rdf:resource="#TransitionRule"/>
</rdfs:Class>

```

```

<!-- Mediators -->

```

```

<rdfs:Class rdf:ID="GGMediator"
  rdfs:label="GGMediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="OOMediator"
  rdfs:label="OOMediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="WGMediator"
  rdfs:label="WGMediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</rdfs:Class>

```

```

<rdfs:Class rdf:ID="WWMediator"
  rdfs:label="WMediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</rdfs:Class>

```

```

<rdf:Property rdf:ID="target"
  rdfs:label="target">
  <rdfs:comment>A mediator may have one or more targets.</rdfs:comment>
  <rdfs:domain rdf:resource="#Mediator"/>
</rdf:Property>

```

```

<rdf:Property rdf:ID="usesService"
  rdfs:label="usesService">
  <rdfs:comment>
    A mediator may use a Web service, goal, or wwMediator for its implementation.
  </rdfs:comment>

```

```

    <rdfs:domain rdf:resource="#Mediator"/>
  </rdf:Property>

  <rdf:Property rdf:ID="source"
    rdfs:label="source">
    <rdfs:comment>A mediator may have a source.</rdfs:comment>
    <rdfs:domain rdf:resource="#Mediator"/>
  </rdf:Property>

  <!-- Relating the WSML vocabulary to the OWL vocabulary -->

  <rdf:Property rdf:about="#version">
    <rdfs:subPropertyOf rdf:resource="&owl;AnnotationProperty"/>
    <owl:sameAs rdf:resource="&owl;versionInfo"/>
  </rdf:Property>

  <rdf:Description rdf:about="&owl;imports">
    <rdfs:subPropertyOf rdf:resource="#importsOntology"/>
  </rdf:Description>

</rdf:RDF>

```

Appendix B. Serving WSML specifications on the Web

When publishing ontologies on the Web, the namespace of the ontology need not be the same as the URI where the ontology file is located. A namespace URI should be a *neutral* URI instead, which allows changes in the definition or the language (HTML, WSML or even OWL) in which the ontology is defined, without changing the namespace and thus keeping compatibility. An example *neutral* URI is <http://example.com/ontologies/e-shop> which can be contrasted to the following implementation-specific URI <http://dev.example.com/ontologies/e-shop/v3.14/e-shop.wsml>. Redirection or internal URI rewriting can be used to serve the WSML document from the neutral URI.

The issue of what should be available at the namespace URI is not yet resolved in an agreed standard way, see [W3C Technical Architecture Group issue namespaceDocument-8](#). However, it seems useful to make different content available for different agents, for example a Web browser going to an ontology URI might get the textual HTML page, whereas an automated agent will retrieve the WSML document from the same URI. The separate documents (the HTML description and the WSML ontology) can still have their own separate URIs, but they should be redundantly available at the namespace URI as well.

In [\[HTTPExamples\]](#) the W3C Semantic Web Best Practices and Deployment Working Group drafts a number of techniques for configuring the popular Apache Web server for serving different content to different user agents, depending on the agents' capabilities represented with the set of accepted media types. These tips can also be followed when serving together the HTML description and the WSML ontology in any of the provided syntaxes (surface or WSML/XML) or in the RDF form presented in this document. The media types in the tips must be changed as follows: files in the surface syntax of WSML should be served under the media type `application/x-wsml` and WSML/XML files should have the media type `application/x-wsml+xml`, as specified in [\[WSML\]](#). Finally WSML/RDF documents should be served using the media type `application/rdf+xml` (provided RDF/XML is used to serialize the RDF graph).

References

[RFC4122] P. Leach, M. Mealling and R. Salz. *A Universally Unique Identifier (UUID) URN Namespace*. IETF RFC 4122, 2005. <http://ietf.org/rfc/rfc4122.txt>.

[WSML] I. Toma and N. Steinmetz, editors. *WSML Language Reference*. 2008. WSML Working Draft D16.1v0.3. <http://www.wsmo.org/TR/d16/d16.1/v0.3/>.

[WSML-Semantics] J. de Bruijn, editor. *WSML Abstract Syntax and Semantics*. 2008. WSML Working Draft D16.3v0.3. <http://www.wsmo.org/TR/d16/d16.3/v0.3/>.

[HTTPExamples] A. Miles (editor): *Configuring Apache HTTP Server for RDFS/OWL Ontologies Cookbook*, editor's draft within the Semantic Web Best Practices and Deployment Working Group at W3C; available at <http://www.w3.org/2001/sw/BestPractices/VM/http-examples/>.

Acknowledgements

The work was partly funded by the European Commission under the projects ASG, EASAIER, enIRaF, Knowledge Web, Musing, Salero, Seemp, SemanticGOV, Super, SWING and TripCom; by Science Foundation Ireland under the DERI-Lion Grant No.SFI/02/CE1/I13 ; by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the projects Grisino, RW, SemBiz, SeNSE and TSC.

The editors would like to thank to all the members of the WSML working group for their advice and input into this document.

Footnotes

[1] The part-whole ontology we use has a distinction between `hasPart` and `hasPart_directly`. `hasPart` is a transitive property, and thus we would not be able to infer which part is directly contained in which whole. `hasPart_directly` is a subproperty of `hasPart`; thus, for each pair in the `hasPart_directly` relation is also in the `hasPart` relation, but it allows us to distinguish direct part-whole containment.