



## D32v0.1 WSML/RDF

WSML Working Draft 2 December 2005

**This version**

<http://www.wsmo.org/TR/d32/v0.1/20051202/>

**Latest version**

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

**Editor:**

Jos de Bruijn

**Authors:**

Jos de Bruijn

Jacek Kopecký

Reto Krummenacher

**Reviewers:**

Atanas Kiryakov

Copyright © 2005 DERI ®, All Rights Reserved. DERI liability, trademark, document use, and software licensing rules apply.

## Abstract

We introduce the RDF representation of WSML and show how the ontology component of WSML is layered on top of RDFS. The RDF representation in this document will eventually replace the RDF syntax in the original WSML specification.

---

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

## Table of Contents

- 1. Introduction
- 2. Relation between WSML and RDF
- 3. RDF representation of WSML
  - 3.1. General structure of WSML/RDF graph
  - 3.2. IRIs and data values
  - 3.3. Ontologies
  - 3.4. Goals
  - 3.5. Web Services
  - 3.6. Mediators
- 4. Mapping WSML to RDF
- 5. Conclusions
- Appendix A. An RDFS ontology for WSML/RDF
- Appendix B. Serving WSML specifications on the Web
- References
- Acknowledgements

## 1. Introduction

The original WSML specification in deliverable D16.1v0.21 [WSML] presents an RDF representation for WSML. However, the specification of the RDF syntax in D16.1v0.21 is unsatisfactory for a number of reasons:

- The RDF syntax for WSML was only specified as a mapping from the surface syntax to RDF triples. This mapping is hard to understand. More specifically, this mapping by itself does not allow one to understand a WSML/RDF document.
- No RDF Schema was specified for the RDF representation.
- The reuse of RDF(S) vocabulary is more limited than strictly necessary (e.g., `rdfs:Class` was not used)
- The RDF syntax does not adopt the RDF semantics and it is in general not clear how a WSML/RDF graph interplays with other RDF graphs
- Treatment of anonymous identifiers is incorrect

With this deliverable we aim to create an updated RDF representation for WSML, reusing existing vocabularies, and to clarify the relation between WSML/RDF and other RDF graphs.

This deliverable is structured as follows. We explain the relationship between WSML specifications and RDF graphs in [Chapter 2](#). We present the RDF representation for WSML in [Chapter 3](#). In [Chapter 4](#) we present the formal mapping between WSML surface syntax and WSML/RDF. This mapping specifies RDF graphs corresponding to WSML specifications.

## 2. Relation between WSML and RDF

The Web Service Modeling Language (WSML) is a language for specifying ontologies (including their instances) and for describing various aspects of Web Services. In order to make WSML data part of the Semantic Web, we need an RDF representation, as it is assumed that the Semantic Web will represent data in this format.

WSML data can be translated to RDF in a straightforward way, making WSML an ontology and the WSML data an instance of that ontology. The following listing shows a simple WSML document and how it could be translated into RDF (following [D16.1 v0.21](#)):

```
// trivial WSML ontology and Web service
namespace {ns_ "http://example.com/"}
ontology ns:Transportation
  concept ns:Vehicle

webService ns:TicketService
  capability ns:TicketServiceCapability

// RDF triples representing the above information
ns:Transportation rdf:type wsml:ontology
ns:Transportation wsml:hasConcept ns:Vehicle

ns:TicketService rdf:type wsml:webService
ns:TicketService wsml:useCapability ns:TicketServiceCapability
```

Such direct translation is a useful solution for most parts of WSML data, but it is suboptimal for WSML ontologies and their instances, because the direct translation of WSML to RDF does not indicate that WSML concepts are similar to RDF classes. In fact, the semantics of WSML concepts and RDF classes differ slightly (for example, `rdfs:Resource` is a class that contains all resources, i.e. everything, but there is no such concept in WSML; and `rdfs:Class` contains itself, which cannot be modeled in WSML).

The direct translation approach (taken originally by d16.1) is based on the assumption that WSML must be represented completely and accurately in RDF. This document relaxes this assumption, recognizing that the result harms reusability, and reusability is the main point of WSML/RDF. Therefore in this document we attempt to translate from WSML to RDF only those parts that are expressible in RDFS, trying to capture the intention of a WSML ontology, not all its properties. The main differences between WSML data and the same data after translation to RDF are listed below:

1. WSML allows putting different and independent non-functional properties on entities with the same identifier but different type (concept, instance, attribute etc.), but in RDF non-functional properties are associated to the identifier, so in effect all the entities with the same name share the same non-functional properties. However, when multiple entities have the same identifier, they must be considered the same in some sense already, so this limitation should not be a problem.
2. In the RDF representation we model attribute values directly as triples, and we do the same for non-functional properties. To differentiate between non-functional properties and (functional) attribute values, we mark non-functional properties as members of the class `nonFunctionalProperty`. The following listing illustrates the mapping:

```
// wsml source data
instance instA
  nonFunctionalProperties
    dc:title hasValue "Instance A"
  endNonFunctionalProperties
  attr hasValue "value"

// resulting RDF triples
instA dc:title "Instance A"
instA attr "value"
dc:title rdf:type nonFunctionalProperty
```

This approach has the limitation that one identifier cannot be used as an attribute and as a non-functional property in the same ontology, but we do not expect this to be a problem.

In effect the RDF representation of WSML cannot reliably serve as syntax for WSML, but it is not intended to do so.

### 3. RDF representation for WSML

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

Non-functional properties in WSML are translated to ordinary triples in RDF. Additionally, for each non-functional property *nfp*, the following triple is added (with *rdf* and *owl* denoting the RDF and OWL namespaces, respectively):

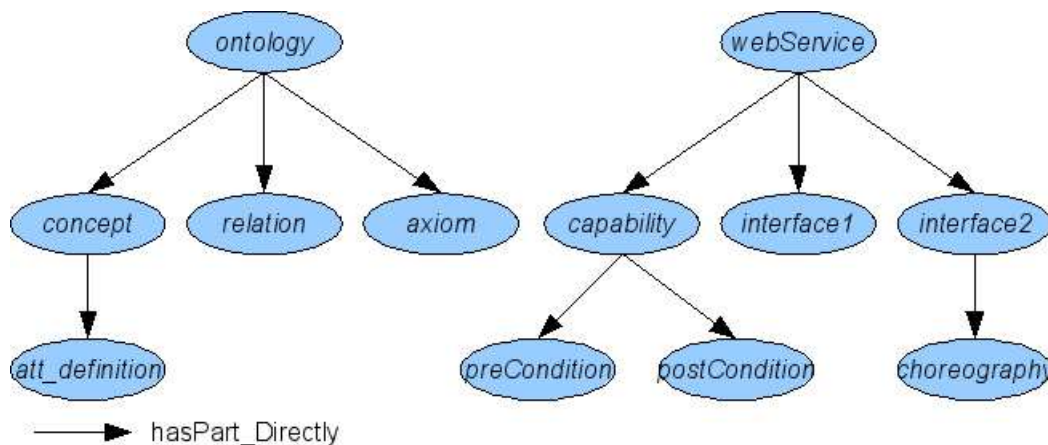
```
(nfp, rdf:type, owl:AnnotationProperty)
```

#### 3.1. General structure of WSML/RDF graph

WSML descriptions group all data related to a particular ontology, web service, goal or mediator in one description. This is achieved in the WSML surface syntax by grouping all descriptions under the **ontology**, **webService**, **goal**, **ggMediator**, **wgMediator**, **wwMediator**, and **ooMediator**. 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 reuse the part-whole ontology developed by the [Semantic Web Best Practices Working Group](http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl): <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl>.

Each ontology, web service, goal, and mediator is a node in the RDF graph, which is connected to all its parts using the relationship `hasPart_directly`.<sup>[1]</sup> This is illustrated in Figure 1.



**Figure 1. WSML Descriptions as part-whole hierarchy**

In some cases it is necessary to disambiguate between different parts of a whole using the `hasPart` property. This is for example the case for the pre-conditions, post-conditions, assumptions and effects of a capability. For example, one axiom may be a precondition of one capability and a postcondition of another capability. In this case, we define sub-properties of `hasPart_directly`, namely `hasPrecondition`, `hasPostcondition`, `hasAssumption`, and `hasEffect`.

Logical expressions are converted to XML literals, according to [\[WSML\]](#), Chapter 9.

#### 3.2. 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 [\[WSML\]](#), [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 translated to new unique identifiers, following the UUID URI schema, as specified in the IETF RFC 4122 [\[RFC4122\]](#).

#### 3.3. Ontologies

Table 1 describes the different types of entities in WSML ontologies. In the table, *wsml* stands for the namespace <http://www.wsmo.org/wsml/wsml-syntax#> and *prefix#localname* stands for the concatenation of the namespace and the local name. 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.

Type	WSML entity	Has parts	Remarks
wsml#ontology	ontology ( <a href="#">2.3</a> )	wsml#concept, wsml#relation, wsml#axiom, wsml#instance, wsml#relationInstance	The WSML variant is indicated using the property wsml#variant .
wsml#concept	concept ( <a href="#">2.3.1</a> )	wsml#attributeDefinition	is a subclass of rdfs:Class
wsml#relation	relation ( <a href="#">2.3.2</a> )	wsml#parameterDefinition	
wsml#instance	instance ( <a href="#">2.3.4</a> )		
wsml#relationInstance	relationInstance ( <a href="#">2.3.4</a> )	wsml#parameterValue	
wsml#axiom	axiom ( <a href="#">2.3.4</a> )		The logical expression, in WSML/XML format, is linked to the axiom via rdfs:isDefinedBy.
wsml#attributeDefinition	( <a href="#">2.3.1</a> )		An attribute definition is part of a concept and has an associated attribute, an possibly range restrictions and min/max cardinalities.
wsml#reflexiveAttributeDefinition	( <a href="#">2.3.1</a> )		A reflexive attribute definition.
wsml#symmetricAttributeDefinition	( <a href="#">2.3.1</a> )		A symmetric attribute definition.
wsml#transitiveAttributeDefinition	( <a href="#">2.3.1</a> )		A transitive attribute definition.
wsml#attribute	( <a href="#">2.3.1</a> )		An attribute is associated with a particular attribute definition.
wsml#parameterDefinition	( <a href="#">2.3.2</a> )		A relation may have a (single) <i>list</i> of parameter definitions, each restricted with the property wsml#ofType or wsml#impliesType .
wsml#parameterValue	( <a href="#">2.3.3</a> )		A relation instance has a <i>list</i> of parameter values, where each parameter value is associated with the actual value using the property wsml#hasValue.

Table 1. Ontologies in WSML/RDF

**Example 2.1** Given the following WSML ontology:

wsmlVariant\_"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

```

namespace { "http://example.org/bookOntology#",
  dc_"http://purl.org/dc/elements/1.1/" }
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 _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 WSMML/RDF representation is the following (using N3 notation):

```

@prefix :
  <http://example.org/bookOntology#>.
@prefix dc:
  <http://purl.org/dc/elements/1.1/>.
@prefix wsmml:
  <http://www.wsmo.org/wsmml/wsmml-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/bookOntology>
  rdf:type wsmml:ontology;
  wsmml:variant, <http://www.wsmo.org/wsmml/wsmml-syntax/wsmml-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 wsmml:concept;
  part-whole:hasPart_directly _:title;
  part-whole:hasPart_directly _:hasAuthor.

_:title
  rdf:type wsmml:attributeDefinition;
  wsmml:forAttribute :title;
  wsmml:ofType xsd:string.

_:hasAuthor
  rdf:type wsmml:attributeDefinition;
  wsmml:forAttribute :hasAuthor;
  wsmml:ofType :author.

:author
  rdf:type wsmml:concept;
  rdfs:subClassOf :person;
  part-whole:hasPart_directly _:authorOf.

_:authorOf
  rdf:type wsmml:attributeDefinition;
  wsmml:forAttribute :authorOf;
  wsmml:ofType :book;
  wsmml:inverseOf :hasAuthor.

:cart
  rdf:type wsmml: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 wsmml:attributeDefinition;
  wsmml:forAttribute :id;
  wsmml:ofType xsd:string;
  wsmml:minCardinality "1"^^xsd:integer;
  wsmml:maxCardinality "1"^^xsd:integer.

_:items

```

```

rdf:type wsm:attributeDefinition;
wsm:forAttribute :items;
wsm:ofType :book.

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

:authorship
rdf:type wsm: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 wsm:parameterDefinition;
wsm:impliesType :author.

_par2
rdf:type wsm:parameterDefinition;
wsm:impliesType :document.

:authorshipFromAuthor
rdf:type wsm: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.

```

### 3.4. Goals

TODO (when there is agreement on the general style of RDF modeling)

### 3.5. Web Services

TODO (when there is agreement on the general style of RDF modeling)

### 3.6. Mediators

TODO (when there is agreement on the general style of RDF modeling)

## 4. Mapping WSML to RDF

This section will contain a mapping from the WSML surface syntax to RDF. The mapping will be created as soon as there is agreement on the RDF representation of WSML. See [Appendix A](#) for the RDF Schema.

## 5. Conclusions

In this first version of this document we have presented an RDF Schema for WSML, as well as a description of the RDF representation of WSML ontologies. The intention of this first version of this document is to serve as a basis for discussion on the WSML/RDF representation, rather than provide a complete mapping.

## Appendix A. An RDFS ontology for WSML/RDF

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

Once the ontology is finalized, it will be accessible at a standardized URI and at <http://www.wsmo.org/wsm/wsm-syntax> for RDF-aware agents.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <ENTITY wsm "http://www.wsmo.org/wsm/wsm-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.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#">
]
```

```
<rdf:RDF xmlns:rdf="&rdf;"
  xmlns:wsm="&wsm;"
  xmlns:rdfs="&rdfs;"
  xmlns:wsmo="&wsmo;"
  xmlns:dc="&dc;"
  xmlns:owl="&owl;"
  xmlns:part-whole="&part-whole;"
  xml:base="http://www.wsmo.org/wsm/wsm-syntax">
```

<!-- 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 SVBP WG): http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#
see also: http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/
```

Non-functional properties are captured using the owl:AnnotationProperty. Each NFP is made  
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.

-->

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

```
<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, instances, axioms and relation instances. A concept (whole) has parts attribute
    definitions.
  </rdfs:comment>
</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.
  </rdfs:comment>
</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.
  </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="mediator"
  rdfs:label="mediator">
  <rdfs:comment>
    A mediator may have a source and multiple targets and may use a service for its implementation.
  </rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="importsOntology"
  rdfs:label="importsOntology">
  <rdfs:comment>
    Any WSML entity (goal, webService, ontology, mediator, interface, capability,
    choreography) may import ontologies in order to reuse vocabularies.
  </rdfs:comment>
  <rdfs:range rdf:resource="#ontology"/>
</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/wsm/wsm-syntax/wsm-full
    http://www.wsmo.org/wsm/wsm-syntax/wsm-flight
    http://www.wsmo.org/wsm/wsm-syntax/wsm-cl
    http://www.wsmo.org/wsm/wsm-syntax/wsm-core
```

In case the same ontology, goal, etc. has different variants associated with it,

```

    the highest variant is chosen.
  </rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="usesMediator"
  rdfs:label="usesMediator">
  <rdfs:comment>Any WSMML 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, a 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="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"/>
  <dc:relation 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"/>
  <rdfs:seeAlso 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
    for ontology, but also, for example, for the specification of the
    functionality of web services through capabilities.

```

The logical expression itself is expressed using the WSMML/XML syntax for logical expressions as an XML literal, which 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 hasPart relationship.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#&rdfs;Class"/>
</rdfs:Class>

<rdfs:Class rdf:ID="instance"
  rdfs:label="instance">
  <rdfs:comment>
    A concept may have a number of instances associated with it,
    through rdf:type.
  </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="parameterValue"
  rdfs:label="parameterValue">
  <rdfs:comment>
    A relation instance has a list of parameter values; each element of the
    list is a parameterValue and is associated to the actual value using
    the property hasValue.
  </rdfs:comment>
</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>
    An RDF property can be seen as a specific kind of WSMML relation, namely a
    binary relation.

    A relation has either an arity or a parameter definition associated with
    it; a relation definition can be compared with the signature specification
    of a predicate in predicate calculus.
  </rdfs:comment>
  <dc:relation>&rdf;Property</dc:relation>
</rdfs:Class>

<rdf:Property rdf:ID="arity"
  rdfs:label="arity">
  <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 has a list of parameter values.
  </rdfs:comment>
</rdfs:Class>

<!-- Capabilities -->

<rdfs:Class rdf:ID="capability"
  rdfs:label="capability">
  <rdfs:comment>
    A web service capability has a number of postconditions,
    preconditions, effects, and assumptions, as well as a list
    of shared variables.
  </rdfs:comment>
</rdfs:Class>

<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 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.
  </rdfs:comment>
  <rdfs:range rdf:resource="#xsd:string"/>
  <rdfs:domain rdf:resource="#capability"/>
</rdf:Property>

<!-- Interfaces -->

<rdfs:Class rdf:ID="interface"
  rdfs:label="interface">
  <rdfs:comment>
    A Web Service interface may have a number of choreographies and one orchestration
    associated with it.
  </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:ID="choreography"
  rdfs:label="choreography">
  <rdfs:comment>
    A Web Service interface may have a choreography associated with it.
  </rdfs:comment>
</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: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="wwMediator">
  <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 service 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 WSMML 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:RDF>
```

## Appendix B. Serving WSML specification on the Web

[Guidelines on how to serve HTML, WSML, WSML/RDF based on requested MIME types; see also <http://www.w3.org/2001/sw/BestPractices/VM/http-examples/2005-11-18/>]

[TODO: next version]

## 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]** J. de Bruijn, editor. *The Web Service Modeling Language WSML*. 2005. WSMO Final Draft D16.v0.21. <http://www.wsmo.org/TR/d16/d16.1/v0.21/>.

## Acknowledgements

The work is funded by the European Commission under the projects DIP, Knowledge Web, InfraWebs, SEKT, SWWS, ASG and Esperanto; by Science Foundation Ireland under the DERI-Lion project; by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the projects RW<sup>2</sup> 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. We would especially like to thank Douglas Foxvog and Eyal Oren for their work on deliverables superseded by this deliverable.

## 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.