



Data, Information and Process Integration  
with Semantic Web Services

**DIP**

*Data, Information and Process Integration with Semantic Web Services*

**FP6 - 507483**

Deliverable

**WP 1: Ontology Reasoning and Querying**

**D1.8**

**WSML Querying**

Atanas Kiryakov

Damyan Ognyanov

Vassil Momtchev

Jul 16<sup>th</sup>, 2006



---

## EXECUTIVE SUMMARY

Here we determine a query language (QL) to be used by ontology management (OM) and semantic-web services (SWS) tools. The language allows tools and applications to efficiently retrieve data from repositories, including reasoners and other tools which store structured data and provide access to it. The proposed approach represents a generic query mechanism for WSML repositories, because:

- DIP has adopted WSMO ([28] and D3.2, [2]) as a SWS modelling framework and WSML as a representation language ([12] and D2.7, [14]);
- There is no query specification for WSML currently available;
- The wide scope of application of this query mechanism renders a DIP-specific proprietary solution inappropriate.

The query mechanism proposed here is usage of the SPARQL query language for RDF, [27], against RDF representation of WSML, according to WSML/RDF, [13]. The main advantages of this approach are as follows:

- SPARQL is the most prominent candidate for Semantic Web query standard. The adoption of existing standard lowers the specification efforts and allows for re-use of software infrastructure, e.g. query parsers and processors;
- High level of interoperability between RDF/OWL and WSML tools is ensured. The mechanism allows that (i) WSML tools retrieve data from non-WSML repositories and (ii) non-WSML tools retrieve data from WSML repositories.

The language does not provide support for data definition and manipulation – those should be handled through the WSMO API (D6.4, [26]).

Querying of WSML repositories with respect to this query mechanism is already supported by ORDI (v.0.4.0, deliverable D2.3 in DIP). Further it is most relevant to the reasoners developed in workpackage WP1 (deliverable D1.9, D1.10, D1.11) and the ontology middleware and the repository developed in WP2 (deliverables D2.4 and D2.5). It is also relevant to all tool and application developers which need to query WSML repositories.

## Document Information

<b>IST Project Number</b>	FP6 – 507483	<b>Acronym</b>	DIP
<b>Full title</b>	Data, Information, and Process Integration with Semantic Web Services		
<b>Project URL</b>	<a href="http://dip.semanticweb.org">http://dip.semanticweb.org</a>		
<b>Document URL</b>			
<b>EU Project officer</b>	Werner Janusch (until Nov 2006 Kai Tullius)		

<b>Deliverable</b>	<b>Number</b>	1.8	<b>Title</b>	WSML Querying
<b>Work package</b>	<b>Number</b>	1	<b>Title</b>	Ontology Reasoning and Querying


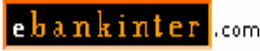


<b>Date of delivery</b>	<b>Contractual</b>	M 24	<b>Actual</b>	31-Dec-05
<b>Status</b>	version. 0.03		final	
<b>Nature</b>	Prototype <input type="checkbox"/> Report <input checked="" type="checkbox"/> Dissemination <input type="checkbox"/> Ontology <input type="checkbox"/>			
<b>Dissemination Level</b>	Public <input type="checkbox"/> Consortium <input checked="" type="checkbox"/>			

<b>Authors (Partner)</b>	Atanas Kiryakov, Damyan Ognyanov, Vassil Momtchev			
<b>Responsible Author</b>	Atanas Kiryakov		<b>Email</b>	<a href="mailto:naso@sirma.bg">naso@sirma.bg</a>
	<b>Partner</b>	Sirma Group	<b>Phone</b>	+359 2 9768 310




<b>Abstract (for dissemination)</b>	A specification of query mechanism for WSML repositories, which is suitable for both ontology management and semantic web services tools. An existing Semantic Web query language (SPARQL) is adapted in combination with RDF representation of WSML.		
<b>Keywords</b>	Ontology querying, WSML, WSMO		

Version Log			
Issue Date	Rev No.	Author	Change
28.Jun.2005	0.01	Atanas Kiryakov	Initial version, the structure is determined
14.Sept.2005	0.02	Atanas Kiryakov	Second version with extensions
22.Nov.2005	0.03	Atanas Kiryakov	Further extensions on the introduction and the main sections of the doc.
16.Jan.2006	0.04	Atanas Kiryakov	
22.Dec.2006	1.0	Atanas Kiryakov	Final version for internal review

## Project Consortium Information

Partner	Acronym	Contact
National University of Ireland Galway	NUIG 	Prof. Dr. Christoph Bussler Digital Enterprise Research Institute (DERI) National University of Ireland, Galway Galway Ireland Email: <a href="mailto:chris.bussler@deri.org">chris.bussler@deri.org</a> Tel: +353 91 512460
Fundacion De La Innovacion.Bankinter	Bankinter 	Monica Martinez Montes Fundacion de la Innovacion. BankInter Paseo Castellana, 29 28046 Madrid, Spain Email: <a href="mailto:mmtnez@bankinter.es">mmtnez@bankinter.es</a> Tel: 916234238
Berlecon Research GmbH	Berlecon 	Dr. Thorsten Wichmann Berlecon Research GmbH Oranienburger Str. 32 10117 Berlin, Germany Email: <a href="mailto:tw@berlecon.de">tw@berlecon.de</a> Tel: +49 30 2852960
British Telecommunications Plc.	BT 	Dr John Davies BT Exact (Orion Floor 5 pp12) Adastral Park Martlesham Ipswich IP5 3RE, United Kingdom Email: <a href="mailto:john.nj.davies@bt.com">john.nj.davies@bt.com</a> Tel: +44 1473 609583
Swiss Federal Institute of Technology, Lausanne	EPFL 	Prof. Karl Aberer Distributed Information Systems Laboratory École Polytechnique Fédérale de Lausanne Bât. PSE-A 1015 Lausanne, Switzerland Email : <a href="mailto:Karl.Aberer@epfl.ch">Karl.Aberer@epfl.ch</a> Tel: +41 21 693 4679
Essex County Council	Essex 	Mary Rowlett, Essex County Council PO Box 11, County Hall, Duke Street Chelmsford, Essex, CM1 1LX United Kingdom. Email: <a href="mailto:maryr@essexcc.gov.uk">maryr@essexcc.gov.uk</a> Tel: +44 (0)1245 436524
Forschungszentrum Informatik	FZI 	Andreas Abecker Forschungszentrum Informatik Haid-und-Neu Strasse 10-14 76131 Karlsruhe Germany Email: <a href="mailto:abecker@fzi.de">abecker@fzi.de</a> Tel: +49 721 9654 0

Partner	Acronym	Contact
Institut für Informatik, Leopold-Franzens Universität Innsbruck	UIBK 	Prof. Dieter Fensel Institute of computer science University of Innsbruck Technikerstr. 25 A-6020 Innsbruck, Austria Email: <a href="mailto:dieter.fensel@deri.org">dieter.fensel@deri.org</a> Tel: +43 512 5076485
ILOG SA	ILOG  Changing the rules of business	Christian de Sainte Marie 9 Rue de Verdun, 94253 Gentilly, France Email: <a href="mailto:csma@ilog.fr">csma@ilog.fr</a> Tel: +33 1 49082981
inubit AG	Inubit  the integration experts	Torsten Schmale inubit AG Lützowstraße 105-106 D-10785 Berlin Germany Email: <a href="mailto:ts@inubit.com">ts@inubit.com</a> Tel: +49 30726112 0
Intelligent Software Components, S.A.	iSOCO 	Dr. V. Richard Benjamins, Director R&D Intelligent Software Components, S.A. Pedro de Valdivia 10 28006 Madrid, Spain Email: <a href="mailto:rbenjamins@isoco.com">rbenjamins@isoco.com</a> Tel. +34 913 349 797
NIWA WEB Solutions	NIWA 	Alexander Wahler NIWA WEB Solutions Niederacher & Wahler OEG Kirchengasse 13/1a A-1070 Wien Email: <a href="mailto:wahler@niwa.at">wahler@niwa.at</a> Tel:+43(0)1 3195843-11
The Open University	OU  The Open University	Dr. John Domingue Knowledge Media Institute The Open University, Walton Hall Milton Keynes, MK7 6AA United Kingdom Email: <a href="mailto:j.b.domingue@open.ac.uk">j.b.domingue@open.ac.uk</a> Tel.: +44 1908 655014
SAP AG	SAP 	Dr. Elmar Dorner SAP Research, CEC Karlsruhe SAP AG Vincenz-Priessnitz-Str. 1 76131 Karlsruhe, Germany Email: <a href="mailto:elmar.dorner@sap.com">elmar.dorner@sap.com</a> Tel: +49 721 6902 31

Sirma AI Ltd.	 <p>                     Sirma  <b>Ontotext</b>  <small>Knowledge and Language Engineering Lab of Sirma</small> </p>	Atanas Kiryakov, Ontotext Lab, - Sirma AI EAD Office Express IT Centre, 3rd Floor 135 Tzarigradsko Chausse Sofia 1784, Bulgaria Email: <a href="mailto:atanas.kiryakov@sirma.bg">atanas.kiryakov@sirma.bg</a> Tel.: +359 2 9768 303
Unicorn Solution Ltd.	 <p>                     Unicorn                 </p>	Jeff Eisenberg Unicorn Solutions Ltd, Malcha Technology Park 1 Jerusalem 96951 Israel Email: <a href="mailto:Jeff.Eisenberg@unicorn.com">Jeff.Eisenberg@unicorn.com</a> Tel.: +972 2 6491111
Vrije Universiteit Brussel	 <p>                     VUB                      Vrije                      Universiteit                      Brussel                 </p>	Pieter De Leenheer Starlab- VUB Vrije Universiteit Brussel Pleinlaan 2, G-10 1050 Brussel ,Belgium Email: <a href="mailto:Pieter.De.Leenheer@vub.ac.be">Pieter.De.Leenheer@vub.ac.be</a> Tel.: +32 (0) 2 629 3749

---

## LIST OF KEY WORDS/ABBREVIATIONS

- Dx.y – the DIP deliverable with the corresponding number;
- RDF – Resource Description Framework, [22] – a basic Semantic Web format for representation of structured data;
- RDFS – Resource Description Framework Schemas, [7] – a schema definition language for RDF;
- SPARQL – a Query Language for RDF, [27] – W3C Candidate Recommendation;
- SWS – Semantic Web Services – semantic descriptions of web services, based on ontologies;
- WSML – Web Service Modelling Language, [12] – a formal language for WSMO;
- WSML/RDF – a mapping of WSML into RDFS, [13];
- WSMO – Web Service Modelling Ontology, [28] – a formalization of the Web Service Modelling Framework, [14], providing a conceptual framework for modelling ontologies and SWS;

## Table of Contents

---

<b>EXECUTIVE SUMMARY</b> .....	<b>I</b>
<b>LIST OF KEY WORDS/ABBREVIATIONS</b> .....	<b>VI</b>
<b>1 INTRODUCTION</b> .....	<b>1</b>
1.1 Query Languages .....	1
1.2 Semantic Repositories .....	1
1.3 Semi-Structured Data .....	2
<b>2 REQUIREMENTS AND DESIGN DECISIONS</b> .....	<b>4</b>
2.1 Design Approach .....	4
2.2 SPARQL as a Basic Language .....	5
2.3 Implications of the Design Decisions .....	5
<b>3 MAPPING WSML ONTOLOGY AND SWS LANGUAGE TO RDFS</b> .....	<b>6</b>
3.1 RDF .....	6
3.2 RDFS and OWL .....	7
3.3 WSMO and WSML .....	9
3.4 WSML/RDF .....	10
3.4.1 Juxtaposing WSML and the tandem of RDF(S) and OWL.....	10
3.4.2 WSML Representation in RDF .....	12
<b>4 SPARQL QUERY LANGUAGE</b> .....	<b>13</b>
4.1 Graph Pattern .....	13
4.2 RDF Dataset .....	13
4.3 Solution Sequence Modifiers .....	14
4.4 Result Form .....	14
4.5 SPARQL Protocol for RDF.....	14
4.6 SPARQL Limitations .....	14
<b>5 REFERENCE IMPLEMENTATION OF WSML QUERYING</b> .....	<b>16</b>
<b>6 CONCLUSION</b> .....	<b>19</b>
<b>REFERENCES</b> .....	<b>20</b>

## 1 INTRODUCTION

This document determines a query language (QL) to be used by tools accessing or managing ontologies and semantic-web service description represented in WSML. The language allows for efficient retrieval of data from repositories, including reasoners and other tools with considerable functionality for storage and retrieval of structured data.

The proposed approach represents a generic query mechanism for WSML repositories due to the following reasons:

- DIP has adopted WSMO ([28] and D3.2, [2]) as a SWS modelling framework and WSML as a representation language ([12] and D2.7, [14]);
- There is no query specification for WSML currently available;
- The wide scope of application of this query mechanism does not allow for a DIP-specific proprietary solution.

In the sub-sections below we provide background information and introduce terms used through the document. The subsequent section 2 presents the different requirements and possible approaches for definition of a query mechanism; then it outlines the approach taken here (namely, SPARQL with respect to the RDF representation of WSML) and some related design choices. Section 3 presents an overview of RDF(S), OWL, WSML and the mapping of WSML to RDF. Section 4 presents SPARQL – the query language adopted here. Section 5 presents a concrete reference implementation of the WSML query mechanism defined earlier, together with relevant examples. The sixth section concludes the document with overview, conclusions, and notes on possible further developments.

### 1.1 Query Languages

The query language proposed here is similar to database query languages, e.g. SQL, [19]. There are number of different tasks related to structured data management, typically performed through such languages – here we mention just three of the major groups:

- Data retrieval – extraction of some data from the repository in a specific format and order and with respect to some constraints. In SQL, data retrieval is performed through **SELECT** queries;
- Data manipulation – updates to the database, including addition, removal, and modification of data. These tasks are handled through the so-called data manipulation languages (DML);
- Data definition – changes to the schemata for structuring of the data. In the case of the relational databases this includes creation and removal of tables, columns, indices, etc. Such tasks are handled through the so-called data definition languages (DDL).

### 1.2 Semantic Repositories

We use the term *semantic repository* here to refer a system for storage, querying, and management of structured data with respect to ontologies. At present there is no single well-established term for such engines. Weak synonyms are: knowledge base

management system, reasoner, ontology server, metastore, semantic/triple/RDF store/database/ repository. Usually, the different wording reflects also differences in implementation, performance, intended application, etc. With the term “semantic repository” we are trying to cover the core functionality offered by most of these tools.

The semantic repositories can be used as a replacement for the database management systems (DBMS), offering easier and more flexible integration of diverse data and more analytical power. In a nutshell, a semantic repository can dynamically interpret metadata schemata and ontologies, which define the structure and the semantics related to the data and the queries. Compared to the approach taken in the relational DBMS, this allows for (i) easier changes to and combinations of data schemata and (ii) automated interpretation of the data. The later means that, for example, given a simple ontology, a semantic repository can return a UK mobile operator, when queried for telecom companies in Europe.

Over the last decade the Semantic Web<sup>1</sup> emerged as an area where the importance of semantic repositories compares to the importance of the HTTP/WWW servers in today’s Internet. This perspective boosted the development, under W3C driven community processes, of a number of robust metadata and ontology standards – RDF(S), OWL and SPARQL being the most basic ones (discussed later on). Those build on top of XML, [6] – the standard which played critical role in boosting the syntactic interoperability. Those play the role, which SQL had for the development and spread of the relational DBMS. Although designed for Semantic Web, these standards face increasing acceptance in wide range of domain – from Enterprise Application Integration (EAI) to life sciences.

### 1.3 Semi-Structured Data

*Semi-structured data* is a term used to refer to two different notions. First, in the text-mining and natural language processing (NLP) communities, semi-structured data are usually considered documents that contain free-text fragments, structured in accordance with some schema. Typical sorts of semi-structured documents are forms and tables, which have some strict structure (fields, sections, etc.), whilst the content of the specific parts of the document is a free-text. Examples are many administrative, insurance, customs, and medical forms.

The second usage of the term “semi-structured” is rather different, denoting a range of more flexible non-relational data-models. The intuition is that, whilst with databases there is a predefined, strict structure of specific tables, fields, and views, there are other, “semi-structured”, representations with less strict structuring, which are still not unstructured<sup>2</sup>. A number of, more or less, graph-based data-models, like RDF, [22], and the Associative Data Model, described in [30], match this understanding of semi-structured data. In both cases, there are two levels of structuring. At the logical<sup>3</sup> level, there is a very simple model, which can be used as a general carrier or canvas for the representation of the data. On top of it, there could be a “softer” and much more

---

<sup>1</sup> <http://www.w3.org/2001/sw/>

<sup>2</sup> See section 3.1.2 of [23] for extended discussion on semi-structured data and its relation to OEM (Object Exchange Model).

<sup>3</sup> With regard to the database terminology.

dynamic schema, which supports the interpretation of the data structured according to the basic model. An intuitive explanation (disregarding the “formal” meaning of “logical model”) is that at the physical level there is a simple and universal data-model (usually some sort of graph), which does not reflect the logical structure, defined through the schemata or ontologies, on top of it.

If we take the latter meaning of “semi-structured”, RDFS, OWL, and WSML are all semi-structured representations. However, we strongly disagree with the philosophy behind this usage of semi-structured. Languages and models like RDF(S) allow dynamic and flexible structuring, which, in our view, is a higher degree of structuring, instead of a “semi”-one.

## 2 REQUIREMENTS AND DESIGN DECISIONS

The query language (QL) used in DIP should be suitable for efficient execution of the following tasks and activities in compliance with WSMO and WSML:

- Ontology management (OM) and in particular retrieval;
- Semantic web service development, management, and execution;
- Data Mediation.

The major requirements towards such query formalism can be summarized as follows:

- R1: Its semantics should have a clear mapping to the one of WSML;
- R2: It should allow for efficient and handy basic data-manipulation. In other words, it should allow for handling of the major tasks performed through SQL towards the relational database management systems (RDBMS);
- R3: It should allow for manipulation of ontologies (schema-level entities like classes, attributes, etc.);
- R4: It should allow for manipulation of SWS entities (WS, Goals, etc.).

R3 is often a subject of discussion regarding general ontology and Semantic Web (SW) query languages. For instance, there is a different level of support for sub-classing in two of the most popular SW languages: RQL, [20], and RDQL, [29]. While RQL provides specific primitives (vocabulary and syntax) related to sub-classes, RDQL does not have any special provisions in this direction. Taking that RDFS schemata are expressed themselves in RDF, in RDQL those can still be referred and used, although the language does not provide specific support. The RQL approach has the advantage that the tools can provide specific (potentially more efficient) support for the schema-related primitives. The advantages of the RDQL approach are simplicity and flexibility; most notably, it is less sensitive towards layering different schemata and epistemology on top of the RDF data-model.

### 2.1 Design Approach

We have considered the following options as an approach for setting up a QL for DIP:

- A1: start from a promising ontology or Semantic Web query language and tune/extend it, if necessary, to properly map to WSML;
- A2: start from WSML, especially its Logical Expressions, and develop a new query language;
- A3: start from DIG-like functional interfaces for reasoners and adapt them to WSML;

After several discussions with the relevant partners in DIP, we reached consensus that approach A1 is most promising. The main reasons were outlined as follows:

- The adoption of an existing standard lowers the specification efforts and allows for re-use of software infrastructure, e.g. query parsers and processors;

- High level of interoperability between RDF/OWL and WSML tools is ensured. The mechanism allows that (i) WSML tools retrieve data from non-WSML repositories and (ii) non-WSML tools retrieve data from WSML repositories.

## 2.2 SPARQL as a Basic Language

There was also a consensus among the involved DIP partners that SPARQL is the Semantic Web language to be adapted as a basis. The advantages can be summarized as follows:

- SPARQL is a next generation Semantic Web Query language, which is addressing a number of known issues with the languages from the previous generations: RQL, RDQL, SeRQL.
- SPARQL has been developed within the RDF Data Access Working Group of W3C, taking into consideration wide range of requirements, including such coming from large industrial members;
- SPARQL is already widely adapted as a recommendation, so, it is very likely to develop into W3C standard;
- This approach covers requirement R2 (basic data manipulation), as long as this is a very strong requirement for it as a Semantic Web QL, as well.

So, finally, the DIP query mechanism was determined to be SPARQL against WSML/RDFS – a WSML representation in RDFS, as discussed in section 3.

## 2.3 Implications of the Design Decisions

SPARQL covers only two out of the four requirements listed above. At present it does not provide support for data manipulation and definition. The intrinsic reason is that, for this sort of activities, it is hard to provide clear definitions of elementary operations, suitable for scenarios with distributed control and responsibility, which are so typical in Semantic Web environment. This difficulty is to a similar degree inherent for the OM and SWS scenarios, relevant for DIP, so, this constraint of SPARQL is natural.

As a workaround, data manipulation should be handled through WSMO API (D6.4, [26]), which defines interfaces for modification of semantic repositories holding WSML (or WSML-compliant) definitions and data.

The problem with the data definition language, which allows for exploration and modification of the schema-level information in a semantic repository, is easier to handle. WSML is a semi-structured language in the sense that there is a no strict separation between data and schema. Thus schema-level definitions, for instance such of concepts, attributes, and web services, can be handled through the standard mechanism for retrieval and management – in this case, SPARQL queries against WSML/RDF and calls through the WSMO API.

Finally, the usage of the WSML/RDF as a bridge between WSML and the RDF data-model for query purposes, added an important usage scenario for this specification. Those requirements were taken into account in the latest revisions of WSML/RDF from year 2006. Please, refer to section 5 for description of a scenario presenting the support of this query mechanism in ORDİ.

### 3 MAPPING WSML ONTOLOGY AND SWS LANGUAGE TO RDFS

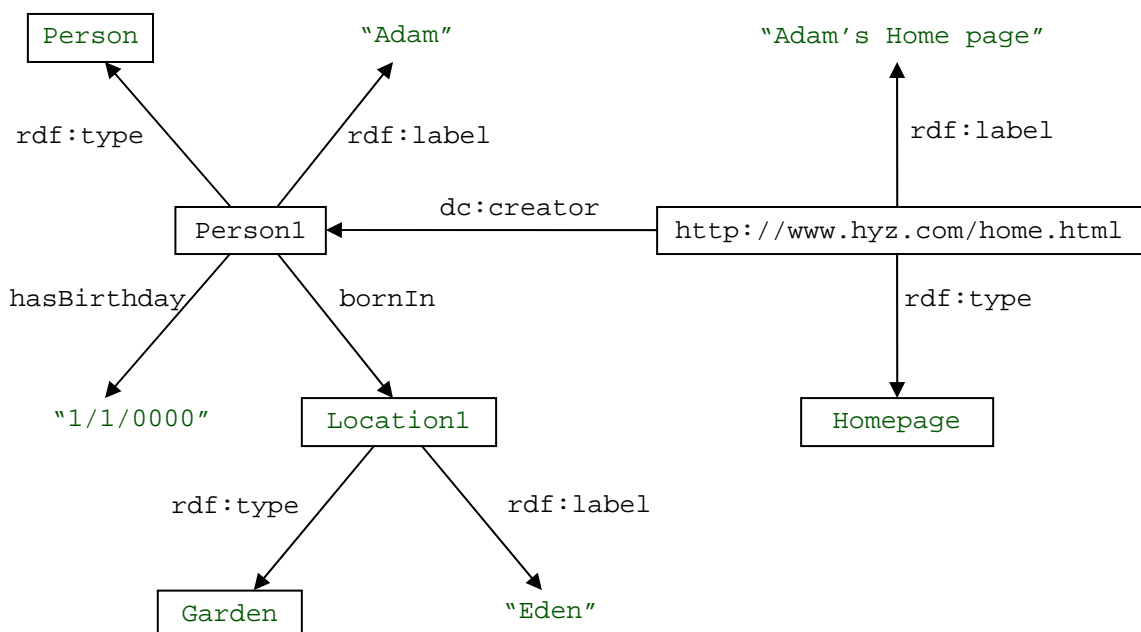
This section starts with a brief introduction to RDF, RDFS, OWL, and WSML – their essentials, variants and dependencies. Good understanding of RDF is crucial for understanding the SPARQL language. RDFS and OWL are used to define the WSML/RDF mapping, which is represented on its own at the end of the section.

#### 3.1 RDF

A family of mark-up and KR standards were developed, under W3C-driven community processes, as a basis for the Semantic Web. RDF, [22], is a metadata representation language, which serves as a basic data-model for the Semantic Web. It allows resources to be described through relationships to other resources and literals. The resources are defined through URIs (unified resource identifiers, [4]; e.g. URL). The notion of resource is virtually unrestricted; anything can be considered as a resource and described in RDF: from a web page or a picture published on a web to concrete entities in the real world (e.g. people, organisations) or abstract notions (e.g. the number Pi and the musical genre Jazz). RDF literals are formally defined as well-balanced, self-contained content of an XML element. Literals are used to represent any concrete data values e.g. strings, dates, numbers, etc. The main modelling block in RDF is the statement – a triple `<Subject, Predicate, Object>`, where:

- **Subject** is the resource, which is being described;
- **Predicate** is a resource, which determines the type of the relationship;
- **Object** is a resource or a literal, which represents the “value” of the attribute.

Fig. 1. RDF Graph Describing Adam and His Home Page



A set of RDF triples can be seen as a graph, where resources and literals are nodes and each statement is represented by a labelled arc (the Predicate or relation), directed from the Subject to the Object. So-called blank nodes can also appear in the graph, representing unique anonymous resources, used as auxiliary nodes. A sample graph, which describes a web page, created by a person called Adam, can be seen in Fig. 1.

Resources can belong to (formally, be *instances* of) classes – this can be expressed as a statement through the `rdf:type` system property as follows: `<resource, rdf:type, class>`. Two of the system classes in RDF are `rdfs:Class` and `rdf:Property`. The instances of `rdfs:Class` are resources which represent classes, i.e. those resources which can have other resources as instances. The instances of `rdf:Property` are resources which can be used as predicates (relations) in triple statements.

The most popular format for encoding RDF is its XML syntax, [3]. However, RDF can also be encoded in a variety of other syntaxes. The main difference between XML and RDF is that the underlying model of XML, [6], is a tree of nested elements, which is rather different from the graph of resources and literals in RDF.

### 3.2 RDFS and OWL

RDFS, [7], is a schema language, which allows for definition of new classes and properties. OWL, [11], is an ontology language, which extends RDF(S)<sup>4</sup> with means for more comprehensive ontology definitions. OWL has three dialects: OWL-Lite, OWL-DL, and OWL-Full. Owl-Lite is the least expressive of these dialects but the most amenable to efficient reasoning. Conversely, OWL-Full provides maximal expressivity but is undecidable<sup>5</sup>. OWL-DL can be seen as a decidable sub-language inspired by the so-called description logics. These dialects are nested such that every OWL-Lite ontology is a legal OWL-DL ontology and every OWL-DL ontology is a legal OWL-Full ontology.

Below we briefly present some of the basic OWL and RDFS modelling primitives:

- All resources, including classes and properties, may have titles (literals<sup>6</sup>, linked through property `rdfs:label`) and descriptions or glosses (literals linked through `rdfs:comment`);
- Classes can be defined as sub-classes, i.e. specializations, of other classes (via `rdfs:subClassOf`). This means that all instances of the class are also instances of its super class. For example, in `city` can be defined as a sub-class of `Location`.
- In OWL, properties are distinguished into object- and data-properties (instances respectively of `owl:ObjectProperty` and `owl:DatatypeProperty`). The object-properties are binary relationships, relating entities to other entities. The data-properties can be considered as attributes – they relate entities to literals.

---

<sup>4</sup> RDF(S) is a short name for the combination of RDF and RDFS.

<sup>5</sup> An undecidable logical language is one for which it is a theoretical impossibility to build a reasoner which can prove all the valid inferences from any theory expressed in that language.

<sup>6</sup> Recall that literals are values such as strings or numbers.

- Domains and ranges of properties can be defined. A domain (`rdfs:domain`) specifies the classes of entities to which this property is applicable. A range (`rdfs:range`) specifies the classes of entities (for object-properties) or data-types of the literal values (in case of data-properties), which can serve as objects in statements predicated by this property. For instance, the property `hasSister` might typically have the class `Person` as its domain and `Woman` as its range. Whenever multiple classes are provided as domain or range for a single property, the intersection of those classes is used.
- Properties can be defined as sub-properties, i.e. specializations of other properties (via `rdfs:subPropertyOf`). Imagine that there are two properties, `p1` and `p2`, for which `<p1,rdfs:subPropertyOf,p2>`. The formal meaning of this statement is that for all pairs for which `p1` takes place, i.e. `<x,p1,y>`, `p2` also takes place, i.e. `<x,p2,y>` is also true. The family relationships can easily provide a number of intuitive examples of sub-properties – for instance, `hasSon` is a special case of `hasChild`, which on its turn is a specialization of `hasRelative`.
- Properties can be defined as a symmetric (via `owl:SymmtericProperty`) and transitive (via `owl:TransitiveProperty`) ones. If `p1` is a symmetric property than whenever `<x,p1,y>` is true, `<y,p1,x>` is also true. If `p2` is a transitive property and `<x,p2,y>` and `<y,p2,z>` are true, it can be concluded that `<x,p2,z>` is also true. `hasRelative` is an example of a property which is both symmetric and transitive.
- Object-properties can be defined to be inverse to each other (via `owl:inverseOf`). This means that if `<p1,inverseOf,p2>` then, whenever `<x,p1,y>` holds, `<y,p2,x>` can be inferred and vice versa. An obvious example is `<hasChild, owl:inverseOf, hasParent>`.

Property restrictions represent a special sort of OWL class definitions. They define a class of all individuals (resources) which satisfy specific conditions regarding a specific property, e.g. cardinality (`owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`) or type of the value (`owl:allValuesFrom`, `owl:someValuesFrom`). While the class descriptions crafted this way are anonymous, they can be used as part of the definitions of named classes as follows:

```
<owl:Class rdf:about="Human">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasParent" />
      <owl:allValuesFrom rdf:resource="#Human" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

This definition states that all instances of the class shall satisfy the constraint, but not vice versa. This means that the restriction represents a partial (necessary, but not sufficient) definition of the class. Complete definitions (including necessary and sufficient conditions) can be crafted using `owl:equivalentClass` instead of `rdfs:subClassOf`.

Restrictions provide a mean for property definitions which are “local” to a class. It is important to realize that `rdfs:range` is determines a “global” restriction on the type of the object of any statements having this property as predicate, disregarding the type of the subject. In the above example, an `owl:allValuesFrom` restrictions is used to say that the parents of human beings are human beings, without putting any constraints on the parents of individuals which are not humans.

### 3.3 WSMO and WSML

WSMO, [28], provides a conceptual model for the description of four types of top-level elements: Ontologies, Semantic Web services, Goals, and Mediators. The description of the top-level elements may contain other types of elements: Concepts, Relations, Instances, Capabilities, etc. The elements could be described by means of non-functional properties (NFP) which represent meta-information such as informal descriptions, reference to the author of the definitions, and other Dublin-Core, [10], alike attributes. Similarly to the annotation properties in OWL, NFP are irrelevant to the formal semantics of the element.

Every WSMO element is identified by one of the following identifiers:

- IRI<sup>7</sup> – Internationalized Resource Identifiers. WSMO elements are denoted by an IRI, except for Literals, Variables (in logical expressions) and anonymous IDs.
- Anonymous IDs, which can be numbered (e. g. `_#1` and `_#2`) or unnumbered (`_#`). The same numbered Anonymous Id represents the same identifier within the same scope (logical expression), otherwise different anonymous IDs represent different identifiers.

As in RDF(S), literals are used to identify values, such as numbers or dates, by means of a lexical representation; they are either plain or typed.

The Web Service Modeling Language (WSML, [14]) enables formal description of all the elements defined in WSMO. WSML is layered into variants (or dialects), corresponding to different levels of logical expressiveness and the use of three different logical paradigms: Description Logics, First-Order Logic and Logic Programming. WSML-Core (the least expressive variant) covers the intersection of Description Logics and Logic Programming and can thus function as the basic interoperability layer between both paradigms. In this respect it is similar to OWL DLP, [18]. WSML-DL (an extension of WSML-Core) is a Description Logic, similar to OWL-DL. WSML-Flight extends WSML-Core in the direction of Logic Programming. WSML-Flight has a rich set of primitives for modeling e.g. value and integrity constraints on attributes, combined with rules. WSML-Flight is designed to allow for efficient decidable reasoning. WSML-Rule extends WSML-Flight to a fully-fledged Logic Programming language, which includes function symbols and puts not restrictions on the use of variables in logical expressions. WSML-Full subsumes both WSML-Rule and WSML-DL; it corresponds to First-Order Logic with non-monotonic extensions.

The base syntax of WSML is the so-called Human Readable (HR) one. WSML HR consists of two parts: (i) the conceptual syntax and (ii) the logical expression syntax.

The conceptual syntax allows for definition of ontologies, goals, web services and mediators – the top-level elements of the WSMO conceptual model. Logical expressions are used to define axioms, using a logical language, which make the definitions of the elements more precise.

The conceptual syntax in WSML HR has a frame-like style, which is closer to the Object-Oriented paradigm, as compared to RDFS. For instance, the definition of a class and its attributes, or an instance and its attribute values, is specified in one large syntactic construct, instead of being divided into a number of atomic chunks (statements). While spreading the information about a particular element across several definitions is possible, it is not “tolerated” by the language.

### 3.4 WSML/RDF

WSML/RDF is the preferred way of representing WSML elements in RDF, defined by the WSML working group in [13]. We will start with a short comparison between WSML and the combination of RDF(S) and OWL. Then we continue with the WSML/RDF representation.

#### 3.4.1 Juxtaposing WSML and the tandem of RDF(S) and OWL

The combination of RDF(S) and OWL provides means for modeling ontologies (schemata) and instance data or metadata structured according to those. A major difference in WSML is that in addition, it provides built-in primitives for modeling of semantic web services (SWS). In this sense, WSML is comparable to the combination of RDF(S) and OWL, on one hand, and a SWS framework such as OWL-S, <http://www.w3.org/Submission/OWL-S/>.

A high-level alignment of the major element types of RDF(S)/OWL and WSML is provided in Table 1. The table does include WSML elements related to web-services, goals, and mediators, as those clearly have no equivalents in RDF(S)/OWL.

**Table 1. Juxtaposing WSML and RDF(S)/OWL Elements**

WSML	RDF(S)/OWL	Comment
Ontology	Ontology	
Concept	Class	
Attribute	Property	The attribute definitions in WSML are local. See the comment further in this section.  In WSML there is no strict distinction in the modelling of object- and datatype-properties; this is determined by the range of the attribute.
Relations	No equivalent	n-ary predicates are not part of the epistemology of RDF(S)/OWL.
Instance	No formal mean; Any Resource	In RDF(S) and OWL, there are no specific primitives for modelling instances (apart from <b>sameIndividualAs</b> and similar in OWL). Instance is every resource which is defined to be of a specific type through:

		<I, rdf:type, C>.
dc:title	rdfs:label	Standard way for providing human-readable names to elements.
dc:description	rdfs:comment	Standard ways for providing description to the definition of a primitive (concept, attribute, etc.)
wsml:subRelationOf	rdfs:subPropertyOf	Those are not direct equivalents, as properties are mapped to attributes, not to relations. Avoid usage of sub-properties or defined the dependency through a WSML axiom
Axiom	No equivalent	
Non-Functional Property	Annotation Property	

There are many parallels and similarities between the languages, which are properly exploited in WSML/RDF in order to provide a proper WSML representation. However, there are few principle differences, which shall be outlined:

- In RDF, every triple can be interpreted separately, while many parts of WSML descriptions are context-dependent, for example, non-functional properties. In WSML, URIs are interpreted depending on their context, because their context is always clear. In RDF, there is no such distinction of context and therefore it is very hard to capture WSML completely in RDF. The WSML/RDF specification solves this problem to some extent by
- One important difference between WSML, on one hand, and RDFS, on the other, is that attributes are defined locally to a class and should not be used outside of the context of that class and its subclasses. In contrast, in RDFS the property definitions are global. For instance, if the property **parent** is defined in RDFS to have domain and range **Animal**, one cannot make a more specific definition for the range of this property for instances of a more specific class, e.g. **Person**. Still, WSML attribute names are global and it is possible to specify global behaviour of attributes through axioms. In this sense, the attribute definitions in WSML are similar to the property restrictions in OWL. However, formally, the attribute definitions are not fully-fledged class definitions, as the property restrictions are in OWL.
- Relations and relation instances have no epistemological equivalents in RDF(S)/OWL. While N-ary predicates can be handled through different patterns (e.g. auxiliary classes), there is no specific support for those. WSML/RDF assures representation of those in RDF, but one should be aware that RDF(S) tools would not be able to map those to higher level notions, i.e. mapping similar to the one of concepts to classes is not present.
- The concept of anonymous IDs in WSML is similar to blank nodes in RDF, [16], however there are some differences. Blank nodes are essentially existential quantified variables, where the quantifier has the scope of one document. In

contrast, the scope of definition of one anonymous ID is a logical expression; they are not existentially quantified variables, but rather local constants.

### 3.4.2 WSML Representation in RDF

The atomic elements in WSML map to the corresponding ones in WSML as follows:

- IRIs remain unchanged. One can choose to alter the name-space of the RDF representation, but this is not required;
- WSML data values map to XML literals in RDF, as defined in [13];
- Anonymous identifiers are translated to new unique identifiers, as defined in [13];

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 it works for Web services, goals and mediators. In WSML/RDF, WSML descriptions are presented as explicit 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): <http://www.wsmo.org/TR/d32/v0.1/part.owl>.

For a detailed definition of the mapping and examples, please, refer to [13].

## 4 SPARQL QUERY LANGUAGE

SPARQL (SPARQL Protocol and RDF Query Language, [27]) is a Semantic Web candidate recommendation presently undergoing standardization by the RDF Data Access Working Group (DAWG, <http://www.w3.org/2001/sw/DataAccess/>) of the World Wide Web Consortium (W3C). As a data access language it is suitable for local and remote usage – SPARQL Protocol for RDF.

SPARQL is a SQL-like query language for getting information from RDF graphs (see section 3.1). SPARQL is a data-access language and protocol and it provides facilities to:

- Extract information in the form of URIs, blank nodes, plain and typed literals.
- Extract RDF sub-graphs.
- Construct new RDF graphs based on information in the queried graphs.

The current SPARQL specification is used only for data-access and do not specify any means to perform data-manipulation of the RDF graph.

The SPARQL Query is composed of 4 different parts: graph/query pattern (GP), RDF Dataset (DS), solution sequence modifiers (SM), and result form (R). It can be presented as a tuple of (GP, DS, SM, R), where the GP and the SM may be empty set. A turtle-like syntax for URIs, QNames and literal is utilized with addition of variables and graph operators to get complex patterns.

### 4.1 Graph Pattern

Graph or query patterns are the triple patterns to be matched against the triples making the RDF graph. The triple patterns to match (**subject**, **predicate**, **object**) are in the form of ([IRI | **variable** | **BNode** | **literal**], [IRI | **variable**], [IRI | **variable** | **BNode** | **literal**]). Literals are allowed in the place of subject, but will fail to match any RDF graph.

Several types of graph patterns exist:

- Basic Graph Patterns, composed only by triple sets, which must all match.
- Group Graph Patterns, formed by graph pattern that must all match or GP fails.
- Optional Graph patterns, additional patterns may extend the solution (similar to left-outer join in SQL).
- Alternative Graph Pattern, where two or more possible patterns are tried (union of patterns).

All the graph patterns may be additionally constrained with Boolean-valued expression or custom application specific function and extended with new keywords.

### 4.2 RDF Dataset

Many applications require the usage of multi-graphs in the RDF data model. The SPARQL query may specify the graph to be used for matching by using RDF Dataset, defines as follows:

$$\{ G, (<U1>, G1), (<U2>, G2), \dots (<Un>, Gn) \}$$

where  $\mathcal{G}$  and each  $\mathcal{G}_i$  are graphs, and each  $\langle u_i \rangle$  is a distinct IRI.  $\mathcal{G}$  is called the default graph. The pairs  $(\langle u_i \rangle, \mathcal{G}_i)$  are called named graphs. SPARQL adds the notion of “default graph” – this is the only difference with the so-called Named Graphs, defined in [8].

If RDF Dataset is not specified the queries are matched against the “default” graph (the merge of all triples). The “named” datasets are identified by unique IRI and scope the search to specific graph only.

### 4.3 Solution Sequence Modifiers

The solution sequence of a query pattern is unordered collection. Sequence modifiers can be applied to the collection as follows:

- Projection (in the **SELECT** clause) can transform one solution and extract only a subset of the variables;
- **DISTINCT** assures that no two answers are the same;
- **ORDER BY** modifier put solution in order of specific column set;
- **LIMIT** defines an upper bound for the number of results;
- **OFFSET** cause the solution to start after a specified number of answers.

### 4.4 Result Form

SPARQL queries support four different result formats:

- **SELECT** returns a SQL-like tabular result compatible with existing protocols like ODBC and JDBC;
- **CONSTRUCT** builds a single RDF graph specified by a graph template;
- **DESCRIBE** is a non-normative description of a named resource, determined by the query service;
- **ASK** probes whether the pattern has a solution without to return the actual result.

### 4.5 SPARQL Protocol for RDF

SPARQL Protocol for RDF, [9], is described in two alternative approaches:

- as a very abstract interface independent of any concrete platform, language, implementation, or binding to another protocol;
- as an HTTP or SOAP binding of this interface associate with specific WSDL and W3C XML Schema documents primary intended to the implementation of SPARQL query services and clients.

One can find the complete definition of the protocol in [9].

### 4.6 SPARQL Limitations

SPARQL has several important limitations compared to the traditional DBMS data-access protocols like ODBC. SPARQL Protocol for RDF is not a state-ful protocol, so the traditional database approach of using cursors with a special isolation and locking

level could not be applied. The query language does not offer functionality to perform data modification operation like **INSERT** or **UPDATE** in SQL. Since SPARQL is based on a closed world of instance graphs in the first place, it will be very useful if aggregate function of type **SELECT COUNT(?x)** would be available.

It is our opinion that the issues which have been considered by DAWG, in the process of the definition of SPARQL, are applicable to WSML in the same way in which they were applicable to an RDF. Thus, the above limitations are natural for a query language for WSML as well.

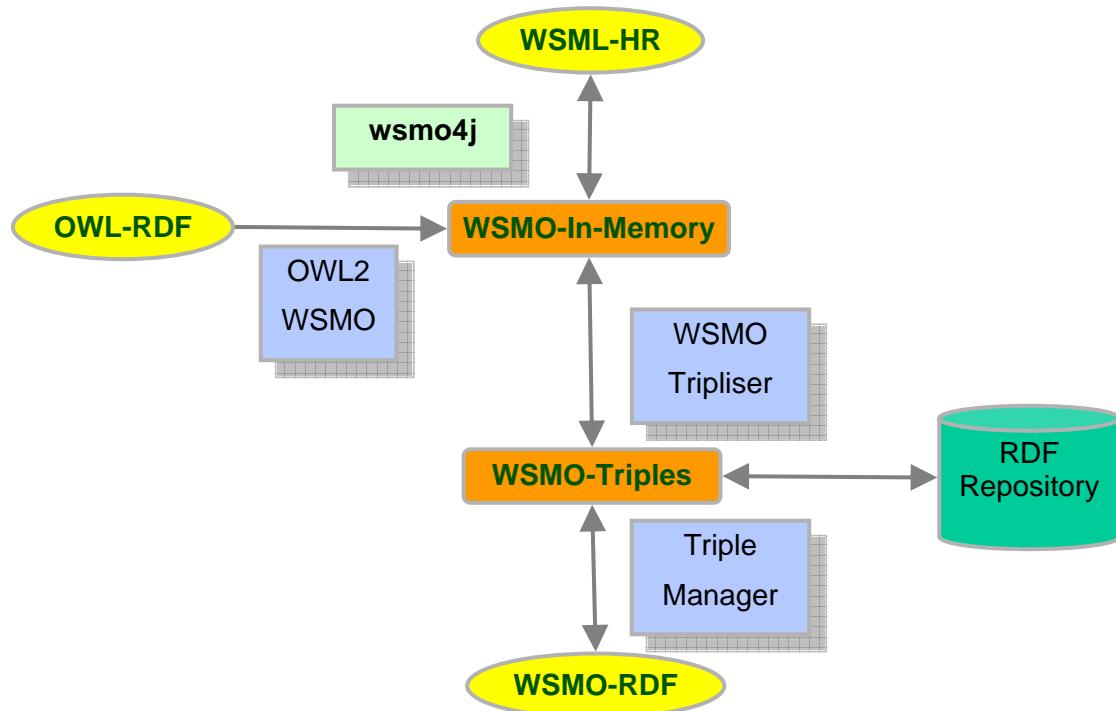
## 5 REFERENCE IMPLEMENTATION OF WSML QUERYING

ORDI is an Ontology Representation and Data Integration conceptual framework, defined in D2.2, [21]. ORDI v0.4 (D2.3, [25]) is an implementation of the framework, which provides two major functionalities:

- a bi-directional WSML to RDF mapping engine and
- high-performance scalable WSML repository.

WSML ontologies and any other elements are translated to RDF following the WSMO/RDF representation (see section 3.4 and [13]). The interoperability between the different formats is presented on Fig. 2. , where WSMO-RDF denotes and RDF serialisation of WSML with respect to the WSML/RDF specification.

Fig. 2. WSML Interoperability with RDF(S) and OWL



The following application scenarios are supported by ORDI:

- Conversion of WSMO objects, accessible through the WSMO API, into WSML/RDF;
- Storage and retrieval of WSMO objects in WSML/RDF format to and from an RDF repository;
- Conversion of WSMO objects from WSML/RDF into WSMO API objects;
- Retrieval of WSML elements by means of SPARQL queries against RDF repository.

These experiments confirmed the feasibility of the overall scheme and provided requirements for the first official version of the WSML/RDF specification, published in

December 2006 and implemented in ORDI v0.4.1, which is available for download at <http://www.ontotext.com/ordi/ordi-0.4.1.zip>.

ORDI comes packed with several sample Java applications, which demonstrate the above scenarios. Follow query example from the `sparqlQueryExample` application. A pre-conditions for it is that an example ontology (available in the distribution at `test/wsm1v2.wsm1`) is loaded in the repository; this can be achieved though the `StoreOntologyExample` application. Excerpts from this test WSM1 ontology follow:

```
wsm1Variant _"http://www.wsmo.org/wsm1/wsm1-syntax/wsm1-rule"
namespace { _"http://www.example.org/ontologies/example#",
            dc _"http://purl.org/dc/elements/1.1#",
            ... }
ontology _"http://www.example.org/ontologies/example"
...
concept Human
...
concept Man subConceptOf Human
...
concept Child subConceptOf Human
...
axiom ChildDef
...
  definedBy
    forall ?x (
      ?x memberOf Human and ageOfHuman(?x,?age)
      and ?age =< 14 implies ?x memberOf Child).

axiom ABoy
  definedBy
    forall ?x (
      ?x memberOf Boy equivalent ?x memberOf Man
      and ?x memberOf Child ) .

instance Paul memberOf { Parent, Man }
  hasChild hasValue George
...
instance George memberOf Man
  hasName hasValue "George Smith"
  hasBirthdate hasValue _date(2004,10,21)
...
```

The following SPARQL query is evaluated against the repository:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX example: http://www.example.org/ontologies/example#
SELECT ?X ?Y
WHERE {?X rdfs:subClassOf example:Human .?Y rdf:type ?X }
```

In a repository with no inference capabilities, it will return the following results:

Man	Paul
Parent	Paul
Man	George

A repository with inference support would also return:

...

Human	Paul
Human	George
Child	George
Boy	George

## 6 CONCLUSION

This deliverable outlined the requirements for a query language for WSML, considering the specifics of the DIP project, but going to a much wider scope of WSMO-aware applications. It defines a generic WSML querying mechanism which can be shortly described as “SPARQL against RDF representation of WSML”. This query mechanism remains neutral to the semantics of WSML, in the same way in which SPARQL is neutral with the semantics of RDF(S) or OWL. There are two major reasons for this:

- Although part of the semantics of WSML can be modelled through mechanisms like query re-writing, this is an implementation specific question instead of an intrinsic requirement to the query language. This is demonstrated by the fact that many of the semantic repositories implement reasoning in a fashion completely independent from the query languages and the corresponding query engines;
- The semantics of WSML varies considerably from one dialect to another. The requirements for the semantics of a query language which combines well with WSML DL are very different from those for a language combining well with WSML Rule, for instance.

The major advantages of the proposed mechanism are obvious:

- The adoption of an existing standard lowers the specification efforts and allows for re-use of software infrastructure, e.g. query parsers and processors;
- High level of interoperability between RDF/OWL and WSML tools is ensured. The mechanism allows that (i) WSML tools retrieve data from non-WSML repositories and (ii) non-WSML tools retrieve data from WSML repositories.

The proposed query mechanism is implemented in the ORDI ontology middleware framework (D2.3), which allows for its straight-forward integration in the components which are already integrated with ORDI: DOME, WSMO Studio, YARS, etc.

There are several prominent directions for the further development of this query mechanism:

- Definition of a transformation procedure, which maps WSML logical expression to **FILTER** clauses in SPARQL. This effort can be combined with an attempt for modelling of fragments of WSML through SPARQL query re-writing;
- Enabling the possibility for retrieval of whole WSML elements, together with their definitions. Although this can easily be allowed through the SPARQL’s **DESCRIBE** queries, a comprehensive description of the expected behaviour is required to make such mechanism useful. Such development can re-use the work performed in ORDI for WSMO “tripleization”.

---

## REFERENCES

- [1] Aduna B.V. (2005). *Sesame RDF Database*. <http://www.openrdf.org>.
- [2] Arroyo, S; Cimpian, E; Dimitrov, M; Domingue, J; Nagypal, G; Spork, M; Vasiliu, L. (2004). *D3.2 Service Description Framework*. DIP Project deliverable. <http://dip.semanticweb.org/>
- [3] Beckett, D. (2004). *RDF/XML Syntax Specification (Revised)*. <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>
- [4] M. Duerst, M. Suignard, Microsoft Corporation, (2005). *Internationalized Resource Identifiers (IRIs)*. Network Working Group, Request for Comments: 3987. <http://www.ietf.org/rfc/rfc3987.txt>
- [5] Biron, P. V; Permanente, P; Malhotra, A; (2004). *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation 28 October 2004. <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- [6] Bray, T; Paoli, J; Sperberg-McQueen, C. M; Maler, E; Yergeau, F. 2006. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, W3C Recommendation 16 August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>
- [7] Brickley, D., Guha, R.V, eds. 2000. *Resource Description Framework (RDF) Schemas*, W3C. <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>
- [8] Carroll, J. J; Bizer, B; Hayes, P; Stickler, P. (2005) *Named Graphs, Provenance and Trust*. WWW2005, <http://www2005.org/cdrom/docs/p613.pdf>
- [9] Clark, K. G (eds.) (2006). *SPARQL Protocol for RDF*. W3C Candidate Recommendation; 6 April 2006. <http://www.w3.org/TR/2006/CR-rdf-sparql-protocol-20060406/>
- [10] DCMI Usage Board. (2003) *DCMI Type Vocabulary*. <http://dublincore.org/documents/2003/11/19/dcmi-type-vocabulary/>
- [11] Dean, M; Schreiber, G. (eds). *OWL Web Ontology Language Reference*. W3C Recommendation, 10 Feb. 2004. <http://www.w3.org/TR/owl-ref/>
- [12] de Bruijn, J; Lausen, H; Krummenacher, R; Polleres, A; Predoiu, L; Kifer, M; Fensel, D. (2005) *D16.1v0.21 The Web Service Modeling Language WSML*. WSML Final Draft 5 October 2005, DERI. <http://www.wsmo.org/TR/d16/d16.1/v0.21/>.
- [13] de Bruijn, J. (ed.); Kopecky, J; Krummenacher, R. (2006) *WSML/RDF*. Deliverable d32v0.1. WSML Final Draft 20 December 2006. <http://www.wsmo.org/TR/d32/v0.1/20061220/>
- [14] de Bruijn, J; Krummenacher, R; Lausen, H; Polleres, A; Predoiu, L; (2005) *D2.7 Ontology Representation Language*. DIP project deliverable: a prototype fact sheet, 04 January 2006. <http://dip.semanticweb.org/documents/Deliverable2.7-final.pdf>.
- [15] Fensel, D.; Bussler, C. (2002). *The Web Service Modeling Framework WSMF*, Electronic Commerce Research and Applications, 1(2), 2002.
- [16] Hayes, P. (2004). *RDF Semantics*. W3C Recommendation, 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>
- [17] HP Labs, Bristol. *Jena RDF(S) API*. <http://www.hpl.hp.com/semweb/jena.htm>
- [18] Grosz, B; Horrocks, I; Volz, R; Decker, St. (2003). *Description Logic Programs: Combining Logic Programs with Description Logic*. In Proc. of WWW2003, Budapest, May 2003.
- [19] ISO JTC 1/SC 32. (2003). *SQL:2003 specification*. ISO/IEC 9075. <http://www.iso.org/>
- [20] Karvounarakis, G; Alexaki, S; Christophides, V; Plexousakis, D; Scholl, M. (2002). *RQL: A Declarative Query Language for RDF*. In Proc. Of WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA. ACM 1-58113-449-5/02/0005.

- 
- [21] Kiryakov, A; Ognyanov, D; Kirov, V. (2004). *D2.2: An Ontology Representation and Data Integration (ORDI) Framework*. DIP project deliverable. <http://dip.semanticweb.org>.
- [22] Klyne, G; Carrol, J. J; (eds). (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation 10 Feb. 2004. <http://www.w3.org/TR/rdf-concepts/>
- [23] Martin-Recuerda, F; Harth, A; Decker, S; Zhdanova, A; Ding, Y; Stollberg, M. 2004. *Deliverable D2.1 "Report on requirements analysis and state-of-the-art"* within WP2 "Ontology Management" of the DIP project. <http://dip.semanticweb.org/>
- [24] Motik, B., Sattler, U., Studer, R. (2005). *Query Answering for OWL-DL with Rules*. *Journal of Web Semantics* 3 (2005), pp. 41-60.
- [25] Ognyanov, D; Kiryakov, A. (2006). *D2.3 Ontology Representation and Data Integration (ORDI) Framework*. DIP project deliverable: a prototype fact sheet, 04 January 2006. <http://dip.semanticweb.org/>
- [26] Ontotext Lab. (2006). *wsmo4j*. An API and a reference implementation for building Semantic Web Services applications compliant with WSMO, <http://wsmo4j.sourceforge.net/>. Earlier version available as DIP project deliverable D6.4, Dec. 2004. <http://dip.semanticweb.org/>.
- [27] Prud'hommeaux, E; Seaborne, A. (eds.) (2006). *SPARQL Query Language for RDF*. W3C Candidate Recommendation; Working Draft 4; October 2006. <http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/>
- [28] Roman, D; Lausen, H; Keller, U. (eds.) 2005. *Web Service Modeling Ontology (WSMO)*, WSMO deliverable D2v1.2. WSMO Final Draft, 13 April 2005. <http://www.wsmo.org/TR/d2/v1.2/>
- [29] Seaborne, A. (2004). *RDQL – A Query Language for RDF*. W3C Member Submission 9 January 2004. <http://www.w3.org/Submission/RDQL/>
- [30] Williams, S. (2002). *The Associative Model of Data*. Second Edition, Lazy Software Ltd. ISBN: 1-903453-01-1. <http://www.lazysoft.com>