



DIP D2.3: Ontology Representation and Data Integration (ORDI) Framework

Prototype Fact Sheet, 30 Jun 2006

This version:

<http://www.ontotext.com/ordi/v0.4/FactSheet.html>

Latest version:

<http://www.ontotext.com/ordi/v0.4/FactSheet.html>

Previous version:

<http://www.ontotext.com/ordi/v0.3/FactSheet.html>

Authors:

Damyan Ognyanov, Ontotext Lab., Sirma Corp., damyan@sirma.bg
Atanas Kiryakov, Ontotext Lab., Sirma Corp., naso@sirma.bg

Reviewers:

Stephan Grimm, FZI
Jan Henke, DERI

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

Copyright © 2006 by [DIP](#). All Rights Reserved. [DIP](#) liability, trademark, document use, and software licensing rules apply.

Document Information

IST Project Number	FP6 – 507483	Acronym	DIP
Full Title	Ontology Representation and Data Integration (ORDI) Framework		
Project URL	http://dip.semanticweb.org		
Document URL	http://www.ontotext.com/ordi/v0.4/FactSheet.html		
EU Project Officer	Kai Tullius		

Deliverable Number	2.3	Title	
Work package Number	2	Title	Ontology Managment

Date of Delivery	contractual	M30	actual	30-June-2006
Status	version	0.4	final	
Nature	Prototype <input checked="" type="radio"/> Report <input type="radio"/> Dissemination <input type="radio"/> Ontology <input type="radio"/>			
Dissemination Level	Public <input checked="" type="radio"/> Consortium <input type="radio"/>			

Authors	Damyan Ognyanov (Ontotext Lab.), Atanas Kiryakov (Ontotext Lab.)		
Responsible Author	Damyan Ognyanov	Email	damyan@sirma.bg
Partner	Ontotext Lab.	Phone	

Version Log			
issue date (dd-mm-yy)	revision no.	author	change
26-06-05	001	Atanas Kiryakov	first internal version (version 1.0)
04-01-06	002	Damyan Ognyanov	second internal version (version 2.0)
09-06-06	003	Damyan Ognyanov	final version for internal review
30-06-06	004	xxx	final submitted version (version 3.0)

Reviewer Information

1	Stephan Grimm		Email	Grimm@fzi.de
	Partner	FZI	Phone	+49 (0)721 9654 816
2	Jan Henke		Email	jan.henke@deri.org
	Partner	DERI	Phone	+43 512 507 6451

Table of contents

[1. Availability and Contacts](#)

[2. Purpose and Functionality](#)

[2.1. Architecture overview](#)

[2.2. wsmo4j and ORDI](#)

[2.3. Related Syntaxes](#)

[2.4. Related Data-models and Representations](#)

[2.5. The Current Version](#)

[3. Requirements](#)

[4. Licensing](#)

[4.1. ORDI License Agreement](#)

[4.2. Licensing of Third Party Libraries](#)

[5. Installation and Usage](#)

[5.1. Installation of ORDI](#)

[5.2. Usage Examples](#)

[6. Future Plans](#)

[References](#)

1 Availability and Contacts

Version: 0.4, 30 Jun 2006.

Download: <http://www.ontotext.com/ordi/ordi-0.4.zip>

Source control: Available from CVS of the [DOMÉ](#) SourceForge project.

Contact person: Damyan Ognyanov, damyan@sirma.bg

2 Purpose and Functionality

Ontology Representation and Data Integration (ORDI) Framework is developed after the analysis and design guidelines of [\[ORDI-Design\]](#) - a conceptual framework, presented in deliverable D2.2 of the DIP project. The major objectives of ORDI are:

- Ontology language neutrality;
- Integration of databases and other structured data-sources;
- Ontology and data modularization;
- Support for heterogeneous reasoners and data-sources.

Instead of developing a new language-independent representation, the implementation of ORDI adapts WSML Core ([\[wsmo0.2\]](#)) as a formal data- and knowledge representation model. This decision was taken due to the following reasons:

- WSML Core is rather close to the model defined in [\[ORDI-Design\]](#). WSML Core is crafted after the same objectives - as a minimal but sufficient basic model, which can serve as a ground and/or be aligned to the models used under the most popular knowledge representation and conceptual modeling paradigms.
- WSML provides a model also for web services (WS), which allows for smooth integration between ontology management (OM) and WS software infrastructure.

ORDI, as a package, contains the following modules:

- **ORDI API** - all the APIs necessary to work with ORDI, at present it is a tiny extension of the WSMO API, see below.
- **ORDI Implementation** - an implementation of the interfaces with the following major parts:
 - A default repository implementation, based on Sesame. It uses the WSMO Tripliser, see below;
 - An RDF/XML Parser implementation for WSMO-RDF, see below;
 - An RDF/XML Parser implementation for import of OWL (RDF/XML syntax). It allows import of OWL through parsing of the most popular RDF/XML syntax and transformation into WSMO-Triples format (see below). Starting with version 0.3 this feature is not part of ORDI; it has been moved to the wsmo4j codebase.
 - A [SPARQL](#) query evaluation module.

Some sample usage code is also included in package, see the [Usage](#) section.

2.1. Architecture overview

ORDI defines higher level repository interfaces and in particular ones for triple based data stores. It comes with a set of modules for triple manipulation allowing storage and retrieval of WSMO entities represented as triples.

A developer who wants to make use of an external repository is required to build a wrapper for it implementing the interfaces presented in ORDI. The main interface is a `TripleStore` defining methods for storing and retrieving of sets of triples. The results are retrieved using iterators allowing streaming and delayed evaluation. This approach is suitable for efficient implementation of query modules. The other significant task will be to translate the data to and from WSMO-API data model and the tool's own proprietary data representation.

The SPARQL query module, part of ORDI, rely on the `TripleStore` interface to evaluate the queries and can be used only with repositories that implement it. A JavaDoc system documentation is included in the distribution and contain information for the implemented functional modules and complete description of the interfaces.

By design ORDI extends `WSMORepository` with query evaluation infrastructure, including generic interfaces to represent the query and the results. The query is passed as plain text wrapped in a helper utility class and the results are retrieved with the aid of an iterator as a sets of bindings of the variables used within it.

2.2 wsmo4j and ORDI

ORDI and [wsmo4j](#) were designed to complement each other in the following way:

- `wsmo4j` includes a WSMO representation and management API coupled with a reference implementation. `wsmo4j` defines APIs for management of the WSMO elements as well as basic in-memory implementations. It also defines few functional interfaces: Parser (parsing and serialization), DataStore (storage and retrieval), Locator (mapping of logical to physical addresses). `wsmo4j` includes a parser for the WSML Human Readable syntax (abbreviated as **WSML-HR**).
- ORDI defines interfaces for more advanced repository (storage, query, maintenance) functionality. It is meant to serve as ontology middleware which mediates between a wide range of ontology management tools and applications on the one hand and different sorts of ontology servers and reasoners on the other. ORDI also takes care of interoperability with other (non-WSMx) representation formats and syntaxes.
- ORDI is an extension of `wsmo4j`, while the latter is self-sufficient. The major dependency between the two is the `org.omg.ontology` package, which defines the ontology primitives of WSMO.

Figure 1 depicts the major relationships between `wsmo4j` and ORDI and their positioning wrt WS- and OM-tools.

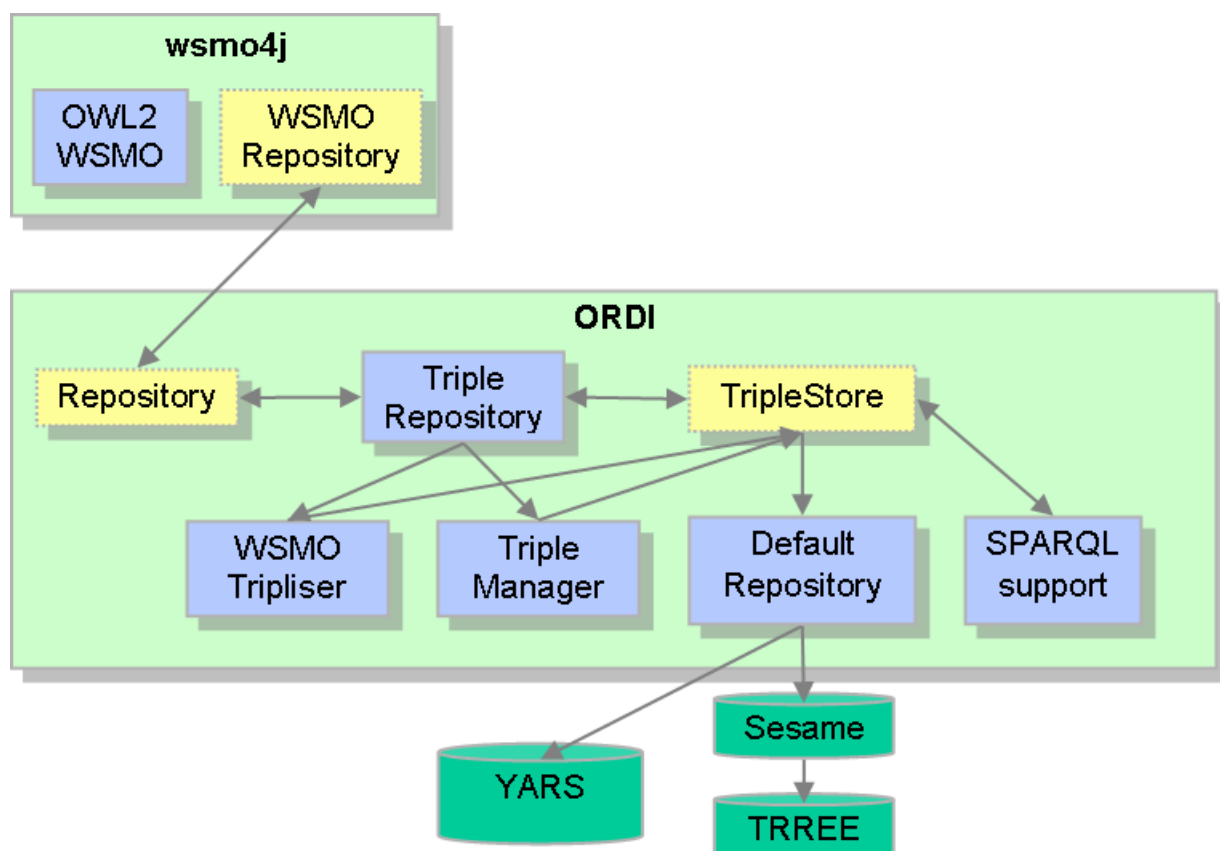


Figure 1. `wsmo4j` and ORDI

2.3 Related Syntaxes

There are numerous file formats which ORDI could process or use via `wsmo4j`. Those will be introduced here, the specific tasks related to them are discussed in a latter sub-section.

- A WSML document in either WSML XML syntax (abbreviated as **WSML-XML**) or WSML Human Readable syntax (**WSML-HR**);
- **OWL-RDF**: the standard RDF XML syntax [\[RDF/XML\]](#). RDF syntaxes different than XML (e.g. NTriples and N3) will also be supported (through the existing RDF parsers). An RDFS subset, which is a proper sub-language of OWL DLP, is considered as an import format.
- **WSMO-RDF**: a WSMO/WSML document serialized according to the [WSML/RDF](#) Schema. The latter is an ORDI-specific RDFS/OWL ontology (meta-schema) derived from the WSML mapping to OWL [\[wsmo0.2\]](#). In a way, it has the same role as the RDFS schema for OWL.

It is important to be mentioned that the immediate plans do not foresee export of WSML into OWL-RDF. The main WSML format compliant with the Semantic Web standards is WSMO-RDF.

2.4 Related Data-models and Representations

There are a couple of datamodels (with corresponding Java interfaces and implementations) relevant to ORDI.

- **WSMO-In-Memory**: a WSMO-API/`wsmo4j` compliant model (e.g. the reference implementation within `wsmo4j`). This is an object-oriented representation, which is not specific for ORDI;
- **WSMO-Triples**: a representation of WSMO elements as RDF triples according to the WSMO RDF Schema. This is an internal representation allowing ORDI to store WSMO entities (and other data) into an RDF triple repository for the sake of efficient query and management of huge amounts of data. The WSMO-RDF syntax is a serialization of this representation.

Here is a diagram which represents the transformations (as gray arrows) between the different formats (depicted by yellow ellipses) and models (depicted by orange rounded rectangles). Next by the arrows one can see the modules which take care of the transformation (depicted by the rectangles).

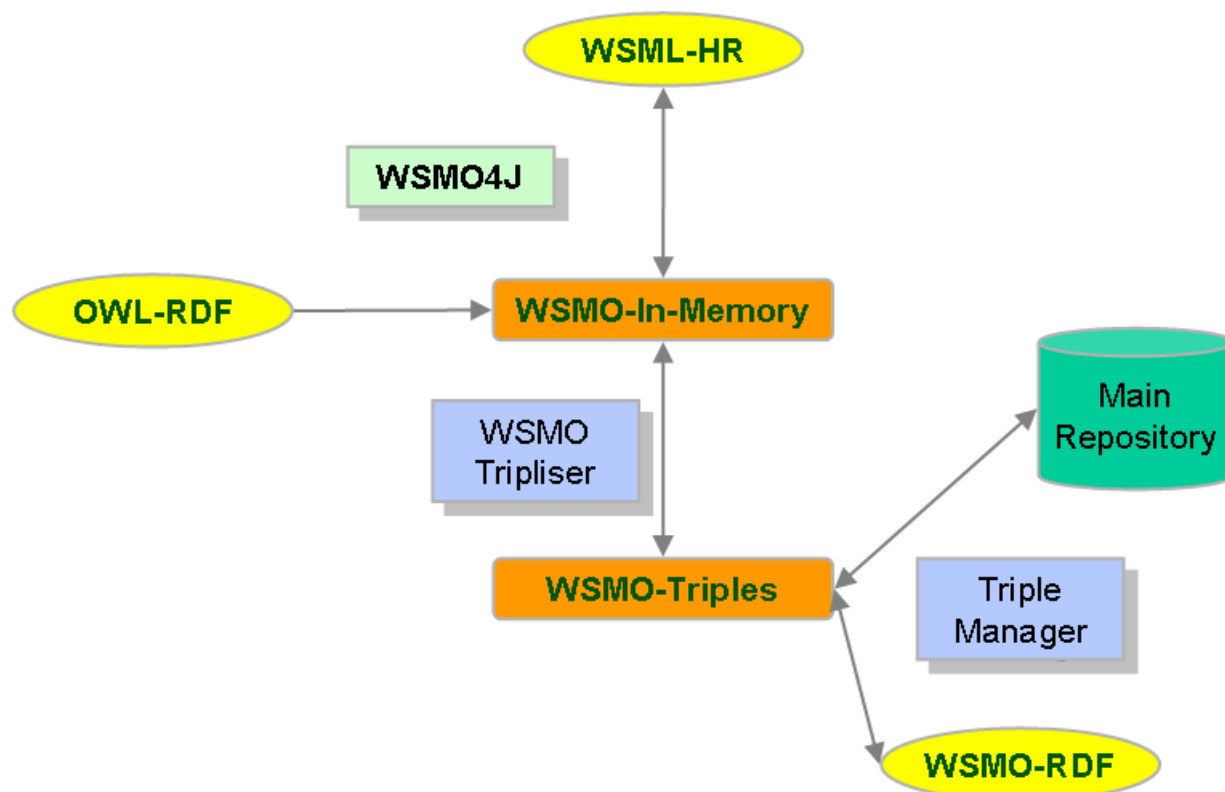


Figure 2. ORDI-related Formats and Representations

2.5 The Current Version

The current version 0.4 of ORDI is updated against the latest release of `wsmo4j`, ver. 0.5.2. It also adapts a new version of the RDF representation of WSML, [\[WSML/RDF\]](#); this version is being finalized in parallel with its implementation in ORDI.

ORDI uses the OWLIM semantic repository (v2.8.3) as a default repository, in order to provide high performance and scalability. OWLIM is a storage and inference layer (SAIL) for [Sesame](#), based on Ontotext's Triple Reasoning and Rule Entailment Engine ([TRREE](#)). OWLIM is proven to scale to tens of millions of statements on desktop hardware; according to the Lehigh University Benchmark ([LUBM](#)) it is the fastest and most scalable OWL repository. Within ORDI, OWLIM is pre-configured to serve as plain RDF repository without reasoning. The usage pattern currently is that ORDI uses Sesame, which uses OWLIM, which uses TRREE. It is planned that in the future ORDI directly will use TRREE.

The other significant change against v. 0.3 is that a query module for [SPARQL](#) was added to ORDI.

The major functionality of ORDI (as added value on top of wsmo4j) is:

- the **more scalable repository** implementation through TRREE;
- the **WSMO-RDF parser, serializer and query facility**

3 Requirements

Nature: A Java library without user interface.

Interfaces (API, Web Services): a Java API.

Platform: JDK 1.5.

Supported standards:

- ORDI's native data-model is the one of wsmo4j, which means [\[wsmo1.2\]](#) as a conceptual model and [\[wsml0.2\]](#) as knowledge representation language.
- ORDI supports export in [\[rdf\]](#), more precisely the [\[RDF/XML\]](#) syntax, which is implemented through Sesame.

Required Libraries ([OMWG](#), [WSMO](#)-related):

- [wsmo4j](#) is an API and a reference implementation for building Semantic Web Services applications compliant with the Web Service Modeling Ontology ([WSMO](#)). The version of wsmo4j used in the current version of ORDI is 0.5.2, which is compliant with [\[wsmo1.2\]](#) and [\[wsml0.2\]](#), from 28-Mar-2006 or newer.

Required Libraries (others):

- [Sesame](#) – Sesame is an open source Java framework for storing, querying and reasoning with RDF and RDF Schema. One of its functionalities is that it can serve as a scalable high-performance semantic repository - this is its major role within ORDI. The version Sesame used for the default triple store in the current version of ORDI is 1.2.4.
- [OWLIM plugin for Sesame](#) – OWLIM is a Sesame wrapper for the TRREE engine that does inference by forward chaining of entailment rules, defined as triple patterns with variables. The version of OWLIM included in the package is 2.3 and it is compatible with Sesame 1.2.4
- [SPARQL Engine](#) - A reference implementation of SPARQL, a current working draft of the W3C, Its aim is a working interface for persistence implementations in addition to an implementation for Sesame RDF store. The SPARQL query support in ORDI uses this library through Sesame but it is incompatible with Sesame 1.2.4 and integrated with sesame-2.0-alpha-3. so both versions of Sesame are required.

4 Licensing

4.1 ORDI License Agreement

Copyright (c) 2005-2006, Ontotext Lab, Sirma.

This library is free software; you can redistribute it and/or modify it under the terms of the [GNU Lesser General Public License](#) as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

4.2 Licensing of Third Party Libraries

Licensing of third party libraries and components required for ORDI:

- [wsmo4j](#) - (c) Copyright Ontotext Lab, Sirma. It is an open-source library, available under the same [LGPL](#) conditions.
- [Sesame](#) - (c) Copyright Aduna b.v. It is an open-source library, available under the same [LGPL](#) conditions.
- [OWLIM](#) - (c) Copyright Ontotext Lab, Sirma. It is an open-source library, available under the same [LGPL](#) conditions.
- [SPARQL Engine](#) It is an open-source library, available under the same [LGPL](#) conditions.

5 Installation and Usage

5.1 Installation of ORDI

ORDI is distributed as a ZIP archive, which should be extracted in a separate folder. The archive file is originally named [ordi.zip](#) and has the following contents:

- `FactSheet.html` - this document;
- `doc` folder - contains Javadoc documentation;
- `ext` folder - contains all the required libraries (jar files);
- `sparql-lib` folder - contains the libraries for the SPARQL support;
- `src` folder - contains all source files;
- `conf` folder - contains all necessary configuration files (at present only the one for Sesame, which is used for the implementation of the default TripleStore);
- `test` folder - contains sample data, at present few ontologies in WSML-HR and OWL format;
- `lib/ordiacpi-0.4.jar` - the ORDI API provided as a Java library. It extends the the WSMO API which is part of `wsmo4j`.
- `lib/ordiiimpl-0.4.jar` - the ORDI default implementation provided as a Java library. It includes WSMO-RDF parser, a Sesame-based default Repository (implementation of the WSMO API `WsmoRepository` interface).

To use ORDI as a library (e.g. in embedded mode) from a Java program, one needs the two ORDI jars (`ordiacpi-0.4.jar` and `ordiiimpl-0.4.jar`) plus the ones in the `ext` folder to be included in the CLASSPATH.

5.2 Usage Examples

Several simple scenarios are provided as an illustration of the functionality of ORDI. Those are available as Java sources in the `src\ordieexamples` folder.

- `StoreOntologyExample` - parsing a WSML-HR ontology and storing it in the default repository. In addition it looks up a concept by IRI, which indirectly loads the concept definition from the repository, because it has registered itself as a locator. The concept is stored in the repository again, without a change, to demonstrate that the basic store/load operation of the repository is definition preserving, although the definition is getting transformed from WSMO-in-memory to WSMO-Triples and back.
- `ModifyConceptExample` - loads a concept definition from the default repository, than modifies it and stores it back; finally it loads the concept again to demonstrate that the definition had changed.
- `ExportToWSMORDFExample` - loads an ontology from the repository and exports (serializes) it into WSMO-RDF format.
- `ParseWSMORDFExample` - parses a previously exported into WSMLRDF ontology. It constructs an in-memory object model. This allows for manipulation and storing the contents of the ontology into an ORDI repository.
- `SparqlQueryExample` - initializes a repository and executes a SPARQL query which is used to explore and/or retrieve specific parts of an ontology, stored previously into the repository. The example shows the results of two SPARQL queries. The first one retrieves all subconcepts of concept `Human` and all the instances of those concepts. The second query retrieves the IDs of all entities that have `dc:description` property with some value and the respective values.

A pre-condition for the second, third and fifth examples is that the `http://www.example.org/ontologies/example` ontology is already stored in the default ORDI repository (which is the effect of the first example: `StoreOntologyExample`).

6 Future Plans

The major driving forces for the future development of ORDI:

- support for the evolution of the related standards and tools (`wsmo4j`, WSMO, WSML);
- developments related to integration of ontology back-end infrastructure, i.e. reasoners and repositories;
- improvements and fixes required by applications using ORDI.

Below follows a non-exhaustive list of tasks, which fit into the short-term development plans:

- Provide WSMO Core reasoning on top of the TRREE engine.
- Provision of a client/server version and database integration.

References

[OWL] McGuinness, D. et al. (2004). *OWL Web Ontology Language*. W3C Recommendation, <http://www.w3.org/TR/owl-features/>

[ORDI-Design] A. Kiryakov, D. Ognyanov, and V. Kirov: *A Framework for Representing Ontologies Consisting of Several Thousand Concepts Definitions*. DIP Project Deliverable D2.2, June 2004. <http://dip.semanticweb.org/deliverables/D22ORDIv1.0.pdf>

[RDF] G. Klyne, J. J. Carroll (eds): *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-concepts/>

[RDF/XML] Dave Beckett (editor): *RDF/XML Syntax Specification (Revised)*. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-syntax-grammar/>

[SPARQL] Eric Prud'hommeaux, Andy Seaborne (eds): *SPARQL Query Language for RDF*. W3C Candidate Recommendation 6 April 2006. <http://www.w3.org/TR/rdf-sparql-query/>

[WSML0.2] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, D Fensel: *The Web Service*

Modeling Language WSMML. Deliverable d16.1v0.2, WSMML, 2005. <http://www.wsmo.org/TR/d16/d16.1/v0.2/>

[WSMO1.2] D. Roman, H. Lausen, U. Keller (eds); J. de Bruijn, Ch. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. Konig-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, M. Stollberg: *Web Service Modeling Ontology (WSMO)*. Deliverable d2v1.2, WSMO, 2005. <http://www.wsmo.org/TR/d2/v1.2/>

[WSML/RDF] Jos de Bruijn (eds); Jos de Bruijn, Jacek Kopecky, Reto Krummenacher: *WSML/RDF*. Deliverable d32v0.1. WSMML Working Draft 15 February 2006 (still, not published). <http://www.wsmo.org/TR/d32/v0.1/20060526/>



\$Date: 2006/20/05 16:34:43 \$