Deliverable 1.2.10

Jens Hartmann (University of Karlsruhe)
Raul Palma (Universidad Politecnica de Madrid)

with contributions from:

Elena Paslaru Bontas (Free University of Berlin)

Abstract.

Most ontologies today exist in pure form without any additional information, e.g. authorship or domain specific information. The proposed Ontology Metadata Vocabulary (OMV) aims to establish a standard which enables users from academia and industry to identify, find and apply – basically meaning to reuse –ontologies effectively and efficiently. We present two complementary reference implementations which show the benefit of such a standard in decentralized and centralized scenarios. The Oyster P2P system for sharing ontology metadata in an early stage of ontology engineering. Well-accepted or recommended ontologies within Oyster are shipped to the up-and-running metadata portal ONTHOLOGY ("anthology of ontologies"), which implements the proposed OMV to support users in accessing and reusing ontologies.
Keyword list: ontology metadata, P2P, repository, reuse

Document Identifier	KWEB/2005/D1.2.10/v1.1
Project	KWEB EU-IST-2004-507482
Version	v1.2
Date	November 20, 2005
State	Final
Distribution	public

OMV Consortium

The Ontology Metadata Vocabulary is based on discussions and agreement among the following consortium.

University of Karlsruhe (UKARL) - Coordinator
Institut für Angewandte Informatik und Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Fax: +49 721 6086580, Phone: +49 721 6086554
Contact person: Jens Hartmann
E-mail address: hartmann@aifb.uni-karlsruhe.de

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Fax: +49 30 83875220, Phone: +49 30 83875223
Contact person: Elena Paslaru Bontas
E-mail address: paslaru@inf.fu-berlin.de

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Fax: +34-913524819, Phone: +34-913367439
Contact person: Raul Palma, Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

Changes

Version	Date	Author	Changes
1.0	20.11.05	Jens Hartmann	creation
1.1	28.11.05	Jens Hartmann	content update
1.2	05.12.05	Raul Palma	content update
1.2	12.12.05	Raul Palma	content update

Executive Summary

Ontologies have seen quite an enormous development and application in many domains within the last years, especially in the context of the next web generation, the Semantic Web. Besides the work of countless researchers across the world, industry starts developing ontologies to support their daily operative business. Currently, most ontologies exist in pure form without any additional information, e.g. missing domain specific or application related information, such as provided by Dublin Core for text documents, denoting the difficulty for academia and industry to identify, find and apply – basically meaning to reuse – ontologies effectively and efficiently. Our contribution consists of a proposal for a metadata standard, so called Ontology Metadata Vocabulary (OMV) and two complementary reference implementations which show the benefit of such a standard in decentralized and centralized scenarios, i.e. the Oyster P2P system and the up-and-running metadata portal ONTHOLOGY ("anthology of ontologies"), which implements the proposed OMV to support users in accessing and reusing of ontologies.

Contents

1 Introduction **1**

2 Requirements **3**

3 Definitions **5**

4 Metadata Organisation and Categorisation **6**
4.1 Property Appropriation 6
4.2 Property Categorisation 7

5 Readers Guide **8**
5.1 Representing a metadata entity 8
5.2 Syntax 8
5.3 Namespace 9

6 Ontology Metadata Vocabulary **10**
6.1 OMV Core vs. Extensions 10
6.2 Conceptualisation vs. Implementation 10

7 Integration **14**
7.1 Knowledge exchange between Oyster and ONTHOLOGY 14

8 Oyster **16**
8.1 Scope 16
8.2 Functionalities 17
8.3 Architecture 19
8.4 Implementation details 20
8.4.1 Installation of Oyster 21
8.4.2 Running Oyster 21
8.4.3 Using Oyster 21
8.5 Preliminary evaluation 23

9 ONTHOLOGY — Ontology Metadata Portal **24**
9.1 Scope 24

9.2 Actors 24
9.3 Functionalities 25
9.4 Architecture 26
9.5 Implementation Details 26

10 Related Work **27**

11 Conclusion **29**

A OMV Core Ontology **32**
A.1 OntologyConceptualisation (OC) 32
A.2 OntologyImplementation (OI) 39
A.3 OntologyType 54
A.4 Individuals 56
A.5 Party 57
A.6 Person 63
A.7 Organisation 66
A.8 LicenseModel 68
A.9 OntologyEngineeringMethod 70
A.10 OntologyEngineeringTool 72
A.11 OntologySyntax 74
A.12 OntologyLanguage 76
A.13 OntologyTask 78
A.14 Location 80

Chapter 1

Introduction

Ontologies are commonly used for a shared means of communication between computers and between humans and computers. To reach this aim, ontologies should be represented, described, exchanged, shared and accessed based on open standards such as the W3C standardized web ontology language OWL. However, most ontologies today exist in a pure form without any additional information about authorship, domain of interest and other meta data about ontologies. Therefore, searching and identifying existing ontologies which are potentially reusable because they e.g. are applied in similar domains, used within similar applications or who have similar properties is a rather hard and tedious task.

We argue that metadata in the sense of machine processable information for the Web[1] helps to improve accessibility and reuse ontologies. Further, it can provide other useful resource information to support maintenance. Thus, we claim that metadata not only help when applied (or, attached) to documents, but also to ontologies themselves.

As a consequence, ontologies which are annotated by metadata require an appropriate technology infrastructure as well. This includes tools and metadata repositories which comply to the ontology metadata standard and which provide the required functionalities to support reuse of ontologies. Such tools and repositories typically should support the engineering process, maintenance and distribution of ontologies.

In this deliverable we present running applications based on the proposed OMV. In detail, we present two complementary applications, namely the decentralised P2P system Oyster and the centralised metadata portal ONTHOLOGY Ḃoth applications have in common that they support single users and communities of users in indentifying, reusing and providing ontology metadata. As a consequence, they support the core idea of the Semantic Web and help to increase the applicability of ontologies. Additionally, we will show how the combined application of both tools will offer users the full potential of ontology metadata management. In general, well accepted ontologies (by quality meassures

[1] `http://www.w3.org/Metadata/`

or human recommendations) are shipped from Oyster into ONTHOLOGY

Chapter 2

Requirements

As an initial step towards a standardized vocabulary, we analysed requirements for ontology metadata. Several aspects are similar to other metadata standards, like Dublin Core. However, important differences like the conceptual models (semantics) behind ontologies require a detailed analysis and require a different representation of metadata about ontologies. In a nutshell, an ontology normally reflects the (i) conceptualization from persons about a specific task or domain which then is (ii) realised by an ontology engineering process [TPSS05] and represented by an ontology document.

As a result, the main identified requirements are the following:

- **Accessibility:** Metadata[1], especially about ontologies, must be accessible and processable for humans and machines.
- **Usability:** a majority of users should be able to apply metadata easily.
- **Reuse of Ontologies:** As ontologies are a core technology for the Semantic Web, its metadata should reflect key issues of the Semantic Web as well. In particular **reuse** and **sharing** of knowledge.
- **Conceptualisation vs. Implementation:** Metadata must reflect (and also distinguish between) a semantic *conceptualisation* and its particular *implementation* as a concrete ontology realisation.
- **Interoperability:** Metadata should be interoperable and conform to the major representation languages currently being used for Semantic Web applications. Indeed, this means that a metadata vocabulary should be representable e.g. in F-Logic and OWL as well.
- **Documentary:** Documentary aspects of metadata like information about *technical, statistical, accessibility, management information, etc.* should be provided.

[1] In the requirements we mean ontology metadata.

- **Extensibility:** Reflecting special user needs, it is required that beyond such standard metadata facts can be added and extended easily.
- **Expressiveness:** Metadata must be expressive enough to represent all desired aspects, as presented above.

The main aspects are *Conceptualisation vs. Implementation* and *Reuse of Ontologies* which should be reflected by any ontology metadata. Already now, it is possible to capture several technical properties of ontologies, like *used syntax* or *number of classes*, almost automatically like realised by [D$^+$04] for example. Besides technical properties which are obviously relevant, there is a strong demand for representing conceptual metadata, like authorship information, categorizations or underlying methodologies. As consequence, representing these issues by a vocabulary requires an expressive language for the metadata itself which makes it impossible to reuse any existing metadata schema.

Furthermore, these requirements provide a complete and easy to apply ontology metadata that can be (semi) automated generated by ontology tools, such as editors and repositories (e.g. Oyster/ONTHOLOGY).

Chapter 3

Definitions

First of all we define our understanding of metadata for ontologies.

- **Metadata** - Relevant information related to existing data.
- **Ontology Metadata** - Metadata which provides information about ontologies according the ontology metadata requirements, cf. 2.
- **Metadata Ontology** - An ontology representing metadata
- **Metadata Entity** - An entity of such a metadata ontology.
- **Metadata Facts** - An ontology property.
- **OMV** - **O**ntology **M**etadata **V**ocabulary. Name of the proposed metadata ontology.

Chapter 4

Metadata Organisation and Categorisation

In this section we present the organisation and categorisation of metadata (entities) to provide a structured overview of the OMV ontology.

4.1 Property Appropriation

We group certain metadata entities as described in the following list.

- **Required** - Such metadata facts are *mandatory*. Any missing fact leads to incomplete metadata ontology.
- **Optional** - Important metadata facts, but not strongly required.
- **Extensional** - Possibility to include specialized metadata entities. Hence, own metadata facts or pre-defined sets like Alignment can be included.

As a result, the representation and usage of *required* and *optional* metadata facts is standardized, as presented in this document. To enable users including additional facts in some cases, we specify the handling of extensions (in a metadata ontology).

Note, that this information is obvious due the `cardinality` statement, where a 1..any statement indicates a *Required* metadata fact, while a 0..any statement indicates a *Optional* metadata fact

4.2 Property Categorisation

We categorize certain metadata entities belonging to a specific context as described in the following list.

1. **General** - Elements providing general information.
2. **Availability** - Information on how to locate the resource.
3. **Applicability** - Details of usability of the resource.
4. **Format** - Information about technical representations.
5. **Provenance** - Elements providing provenance information, like creator information etc.
6. **Relationship** - Elements providing information about relationships between other resources.
7. **Statistic** - Elements providing statistical properties.
8. **Other** - Providing other information which is not covered by the previous one.

Chapter 5

Readers Guide

In the following we give an overview of notations used in this paper.

5.1 Representing a metadata entity

We use following structure to describe metadata entities. Such an entity can be a *class* or *property (DatatypeProperty, ObjectProperty)* of the OMV ontology.

Name of entity	
Name	Name of OMV entity. Note: Case-sensitive.
Type	Type of entity: `class, ObjectProperty` or `DatatypeProperty`.
Identfier	Used Identifier for this entity.
Appropriation	Whether its `required`, `optional` or `extensional`, cf. section 4.1.
Category	Link to the Category the entity belongs to, cf. section 4.2.
Definition	A **short definition** of the purpose, which might be explained more in detail by the `comments` tag.
Domain	Domain of OMV entity.
Range	Range of OMV entity.
Cardinality	Cardinality of OMV entity (MIN:MAX).
OMV version	OMV version in which the entity has been introduced.
Comments	Detailed comments for the entity.

Table 5.1: Exemplary entity representation

5.2 Syntax

We use the OWL syntax to provide examples of ontology metadata.

5.3 Namespace

The OMV ontology uses the following namespaces:

```
owl ="http://www.w3.org/2002/07/owl#"
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#"
omv ="http://omv.ontoware.org/2005/05/ontology#"
```

Chapter 6

Ontology Metadata Vocabulary

In the following, we present the developed Ontology Metadata Vocabulary (OVM).

6.1 OMV Core vs. Extensions

OMV consists of the metadata core which represents the most common classes and properties required as metadata. Due countless applications, tasks and domains of ontologies the core set might be insufficient for all of them. Therefore we propose a flexible and extensible modularisation mechanism, so called OMV Extensions. These extensions are task specialised ontologies itself which reuse the OMV core as base ontology.

6.2 Conceptualisation vs. Implementation

OMV core distinguishes between the **ontology conceptualisation** and an **ontology implementation** as concrete realisation of an ontology. This separation is based on following observation: any existing ontology document has some kind of *core idea* (conceptualisation). From an ontology engineering perspective, initially a person develops such *core idea* of what should be modeled (and maybe how) in his mind. Further, this initial conceptualisation might be discussed with other persons and after all, an ontology will be *realised* using an ontology editor and stored in a specific format. Over time, there might be created several *realisations* of this initial *cconceptualisation* with many different properties, e.g. in RDF(S) [BG04] or OWL [SWM04].

Therefore we distinguish between a *ontology conceptualisation* and an *ontology implementation*:

- **[Ontology Conceptualisation]:** An *Ontology Conceptualisation (OC)* represents the abstract or core idea of an ontology. It describes the core properties of an ontol-

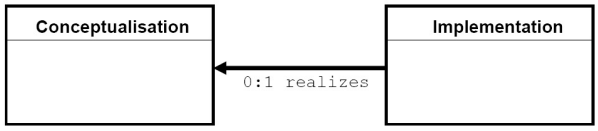

Figure 6.1: Relationship between conceptualisation and implementation

ogy, independent from any implementation details. For a general illustration of the relationship of a conceptualisation and OD, we refer to figure 6.1.

- **[Ontology Implementation]:** An *Ontology Implementation (OI)* represents a specific realization of a conceptualisation. Therefore, it describes properties of an ontology that are related to the realization or implementation.

The distinction between a conceptualisation and OI leads to an efficient mechanism, e.g. for tracking several versions and evolvements of ontologies as well as for different representations of one knowledge model in different languages. In particular, such a *conceptualisation* can be seen as representation of the conceptual model beyond an ontology. Technically, a conceptualisation and an ontology implementation are modeled as two separate classes, with the relation `realizes` from the ontology document to the conceptualisation. This means that there may be many possible ontology implementations for one conceptualisation, but one ontology implementation can only `implement` one conceptualisation.

Normally, an OI should not be able to exist without a corresponding conceptualisation. However, for practical reasons, we allow the existence of each class independently of each other. Hence, we cannot assume that every existing ontology will be annotated by its original author who might create a conceptualisation for his ontology. However, automatically extracting syntactical properties of an existing ontology is quite simple. Then, such *minimal OI* would exist without a concrete conceptualisation.

The main classes and properties of the OMV ontology are illustrated in figure 6.2. Please notice, that not all classes and properties are included. It is only to demonstrate the main idea behind OMV . The ontology is available for download in several ontology formats[1].

It should be noticed that there exist several properties within conceptualisation and OI which look similar at first. However, they have different meanings and semantics. For illustration, think of an ontology engineer `A` developing an ontology in RDF(S) syntax and annotating it with OMV Then, the properties of the conceptualisation and OI individual are quite similar. Exemplary, both would have the same `party` as `creator` and so on. However, over time, there might be another engineer `B` with similar needs according to the conceptualisation from `A`. Hence, `B` reuses the conceptualisation from `A` and only creates

[1]OMV representations are available at `http://ontoware.org/projects/omv/`

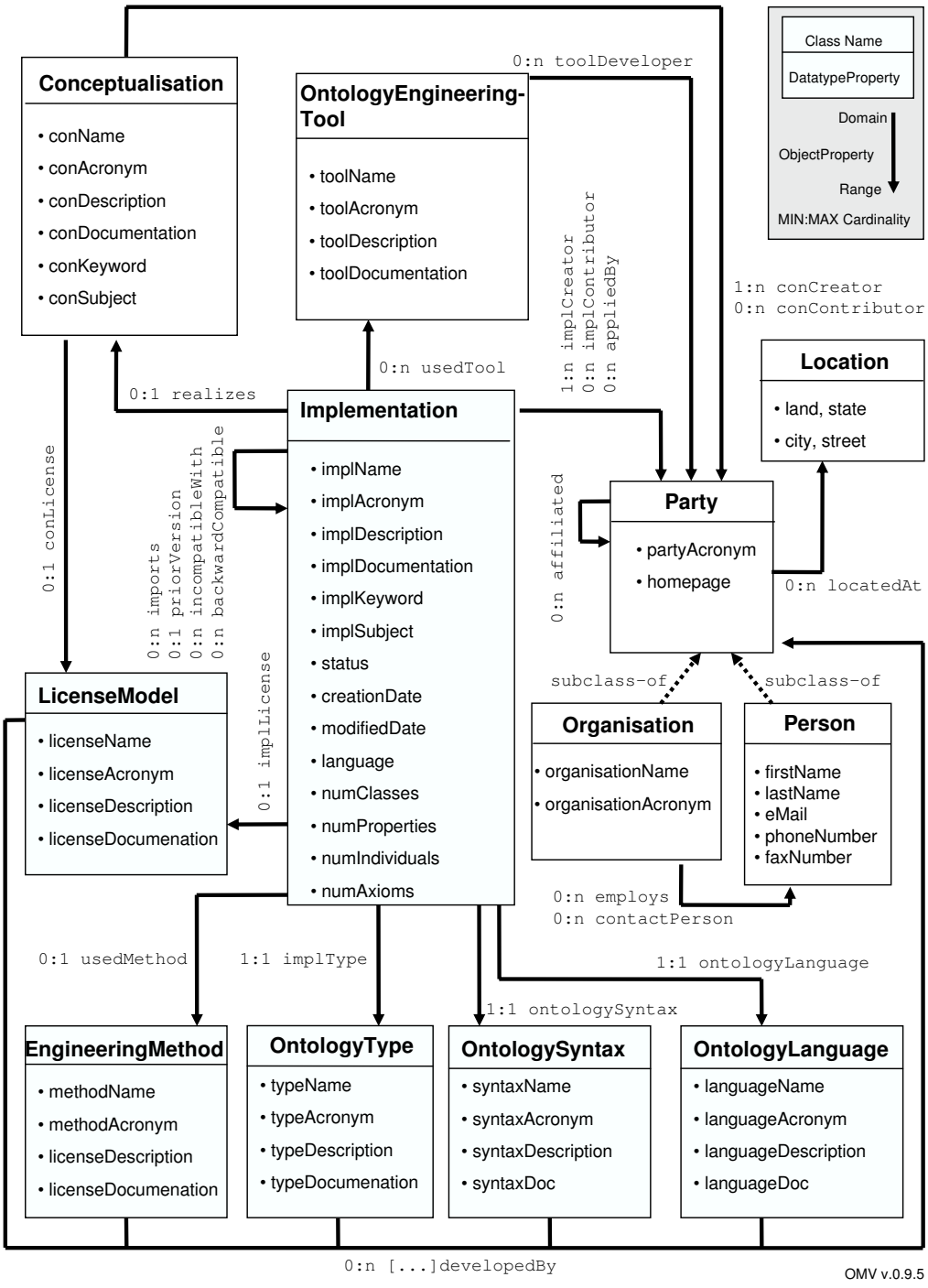

Figure 6.2: General OMV overview

a new OI, e.g. realising the conceptualisation in OWL instead of RDF(S). As a result, a new OI would be created for this ontology and most of the properties are different now.

Chapter 7

Integration

Both applications we present in this deliverable (i.e. Oyster and ONTHOLOGY) are covering a variety of different tasks. Indeed, users who want to store metadata individually similar to managing his personal favorite song list, a repository is required to which a user has full access and can perform any operation (e.g. create, edit or delete metadata) without any consequences to other users. Exemplary, users from academia or industry might use a personal repository for a task dependant investigation or ontology engineers, might use it during their ontology development process to capture information about different ontology versions. We argue that a decentralised system is the technique of choice, since it allows the maximum of individuality while it still ensures exchange with other users. Centralised systems allow reflecting long-term community processes in which some ontologies become well accepted for a domain or community and others become less important. Such well accepted ontologies and in particular their metadata need to be stored in a central metadata portal which can be accessed easily by a large number of users. The benefit of connecting both systems lies mainly in the simple use of existing ontology metadata information within Oyster. So, while users are applying or even developing their own ontologies they can manage their own metadata along with other existing metadata in one application (in Oyster). If some metadata entries from Oyster have reached a certain confidence, an import into ONTHOLOGY can be performed easily. In combination, both systems ensure efficient and effective ontology metadata management for various use cases. The ontologies that will be imported from Oyster to ONTHOLOGY will also have to pass a quality evaluation, which will be defined in the KWeb deliverable 1.2.9.

7.1 Knowledge exchange between Oyster and ONTHOLOGY

Based on recommendations and quality evaluation mechanisms selected ontologies are imported from Oyster into the metadata portal ONTHOLOGY Às exchange mechanism

we rely on the Open Archive initiative[1] which has been successfully applied for a large number of digital libraries among the world. The developed OMV is used as exchange language.

[1] http://www.openarchives.org/

Chapter 8

Oyster

Oyster is a java-based Peer-to-Peer application that exploits semantic web techniques in order to provide an innovative and useful solution for exchanging and reusing ontologies. In order to achieve this purpose, Oyster provides facilities for managing, searching and sharing ontology metadata in a P2P network, thereby implementing the OMV proposal for the standard set of ontology metadata.

8.1 Scope

We argue that Oyster will offer a practical application for exchanging ontology metadata. First, the information sources (ontologies) are geographically distributed among the community, and developers are willing to share the information about the ontologies they created provided they do not have to invest much work in doing so, while at the same time they are able to maintain the ownership of their ontologies. Second, as a Peer-to-Peer system, Oyster further benefits from the following characteristics: no need for a centralized server (thus avoiding a bottleneck for both computational performance and information update), robustness against failure of any single component, and scalability both in data volumes and the number of connected parties. Third, since ontologies can be represented in different languages (such as OWL[SWM04], DAML+OIL[dam01], RDF-S[BG04]), Oyster provides the possibility to exchange heterogeneous information through the use of the metadata standard. Finally, Oyster will provide an overview of the current ontologies available on the community and give an indication of which ontologies become well accepted for a domain or community.

Oyster offers a user driven approach where each peer has its own local repository of ontology metadata and also has access to the information of others repositories, thus creating a virtual decentralized Ontology repository. The Oyster client on its own (e.g. disconnected from the P2P network) will already provide added value to its users as it will give developers an overview and search facilities of his/her own ontology metadata stored

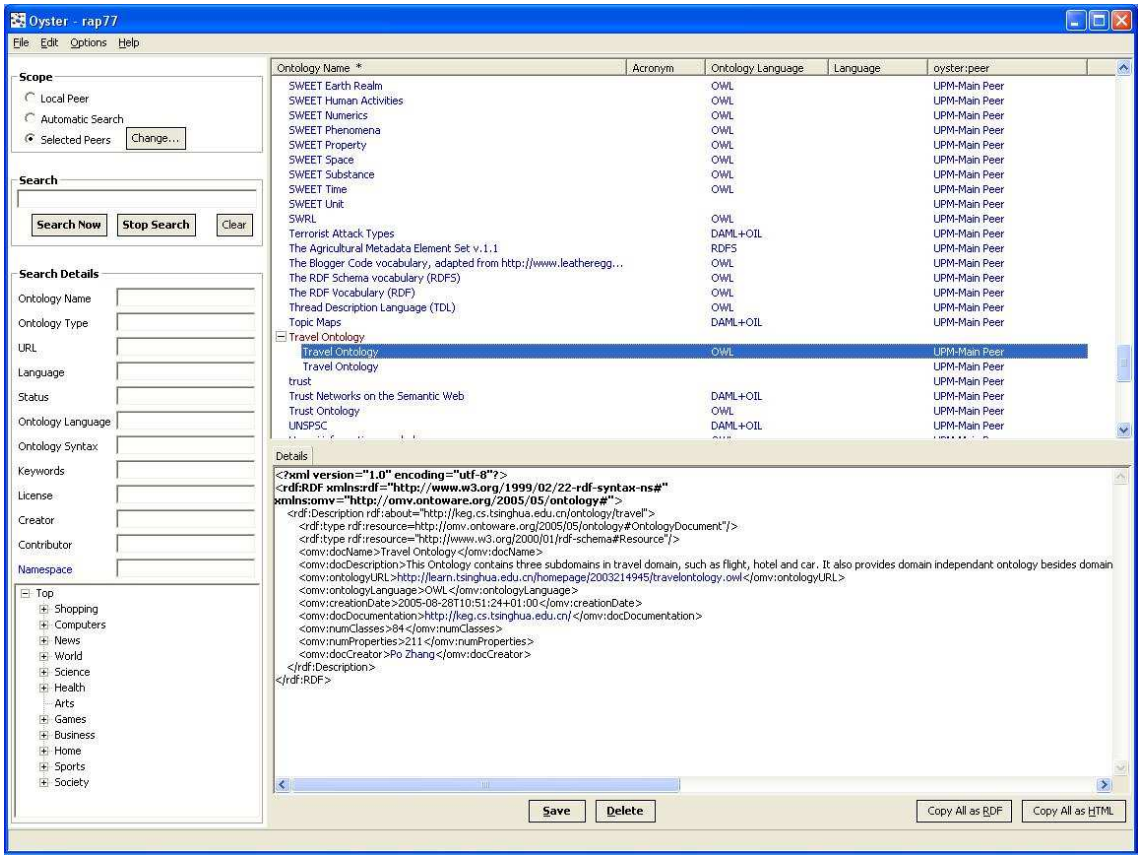

Figure 8.1: Oyster screenshot

in its local repository. The goal is a decentralized knowledge sharing environment using Semantic Web technologies that allows developers to easily share ontology documents.

8.2 Functionalities

The Oyster system has been implemented as an instance of the Swapster system architecture [1]. In Oyster, ontologies are used extensively in order to provide its main functions (importing data, formulating queries, routing queries and processing answers).

Creating and importing metadata: Oyster enables users to create metadata about ontologies manually, as well as to import ontology files and to automatically extract the ontology metadata available, letting the user to fill in missing values. For the automatic extraction, Oyster supports the OWL, DAML+OIL and RDF-S ontology languages. The ontology metadata entries are aligned and formally represented according to two ontolo-

[1] `http://swap.semanticweb.org/`

gies: (1) the proposal for a metadata standard OMV which describes the properties of the ontology, (2) a topic hierarchy (i.e. DMOZ topic hierarchy), which describes specific categories of subjects to define the domain of the ontology.

Formulating queries: As shown in the left pane of the screenshot, the user can search for ontologies using simple keyword searches, or using more advanced, semantic searches. Here, queries are formulated in terms of these two ontologies. This means queries can refer to fields like name, acronym, ontology language, etc. (using the ontology document metadata ontology) or queries may refer to specific topic terms (using the topic hierarchy i.e. DMOZ).

Routing queries: As shown in the upper left pane of the screenshot, the user may query a single specific peer (e.g. their own computer, because they can have many ontologies stored locally and finding the right one for a specific task can be time consuming, or users may want to query another peer in particular because this peer is a known big provider of information), or a specific set of peers (e.g. all the member of a specific organization), or the entire network of peers (e.g. when the user has no idea where to search), in which case queries are routed automatically in the network. In the latter case, queries are routed through the network depending on the expertise of the peers, describing which topic of the topic hierarchy (i.e. DMOZ) a peer is knowledgeable about. In order to achieve this expertise based routing, a matching function determines how closely the semantic content of a query matches the expertise of a peer [HSvH04].

Processing results: Finally, the results matching query are presented in a result list (c.f. upper right pane in the screenshot). The answer of a query might be very large, and contain many duplicates due to the distributed nature and potentially large size of the Peer to Peer network. Such duplicates might not be exactly copies because the semi structured nature of the metadata, so the ontologies are used again to measure the semantic similarity between different answers and to remove apparent duplicates. For Oyster, duplicates are ontology metadata entries which refer to the same ontology, but are modelled as different resources. In order in recognize two entities as being duplicates, Oyster apply specific similarity functions. A similarity function for RDF resources R of the local node repository is a function,

$$sim\text{: R x R} \text{—>} [0..1]$$

with the ontology properties described in OMV.

In detail, we compiled a set of specific ontology facts used to assess the similarity between two instances (e.g. implementationName, implementationType, creationDate, implementationCreator, versionInfo, implementationSubject). For each of these facts we use different individual similarity functions depending on the type of information it represents. Then, from the the results of each individual similarity function, we obtain an overall value with an aggregated similarity function, using a weighted average over the individual functions. These weights were assigned based on experiments with sample

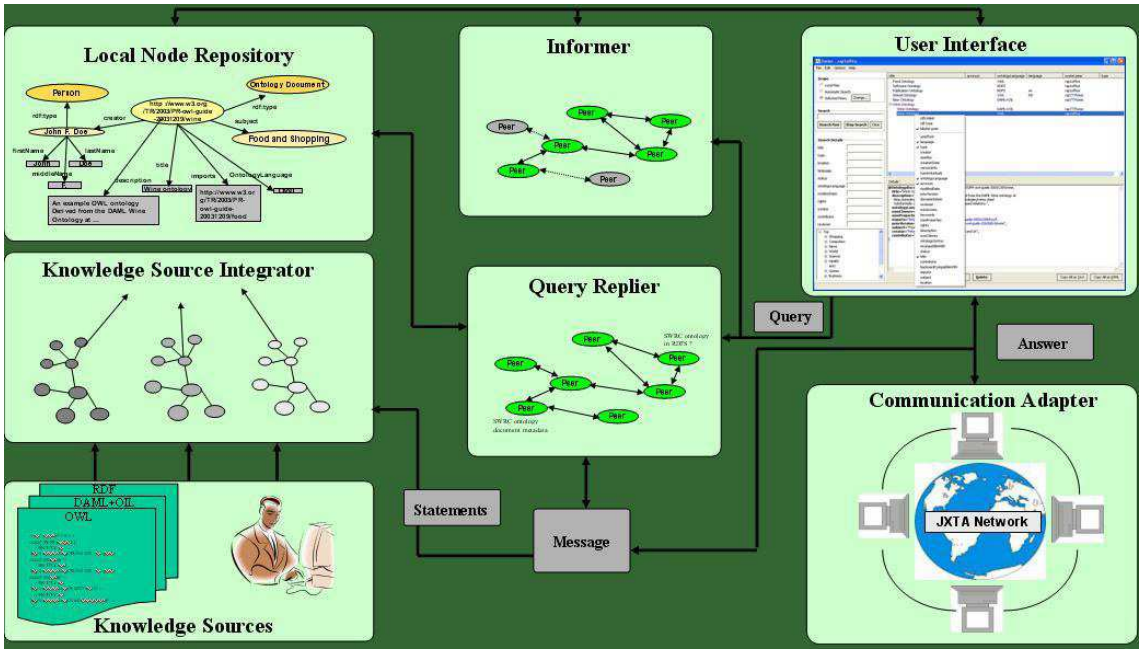

Figure 8.2: Swapster System Architecture

data. Using the overall value, we consider duplicates those pairs of resources whose similarity is larger than a certain threshold.

Then a merged representation that combines the knowledge from the individual and potentially incomplete items is presented to the user. The details of particular results are shown in the lower right of the screenshot. The user can integrate results of a query into their local repository for future use. This information may in turn be used later to answer queries by other peers. Also, as proposed by OMV, all the specific realizations of an ontology can be grouped by the same ontology conceptualisation to organize the answer.

8.3 Architecture

As mention above, Oyster system has been implemented as an instance of the Swapster system architecture as introduced in [BEH+03]. Figure 8.2 shows a high-level design of the archi- tecture of a single node in the Peer-to-Peer system. Briefly, the individual components of a single oyster node are:

Communication Adapter: This component is responsible for the network communication between peers. It serves as a transport layer for other parts of the system, for sending and forwarding queries. It hides and encapsulates all low-level communication details from the rest of the system. In the specific implementation of the Oyster system

we use JXTA [2] as the communication platform.

Knowledge Sources: Sources of ontology metadata, such as Ontology files (e.g. OWL file) from which the ontology metadata can be extracted.

Knowledge Source Integrator: Responsible for the extraction and integration of internal and external knowledge sources into the Local Node Repository. (Extract metadata from ontology file, manage duplicates)

Local Node Repository:

- Mediate between views and stored information
- Support query formulation and processing
- Specify the peer's interface to the network
- Provide the basis for peer ranking and selection
- The Local Node Repository is based on the RDF-S Repository Sesame [3]
- SeRQL [4] is the query language

Informer: Proactively advertise the available knowledge of a peer in the Peer-to-Peer network and to discover peers with knowledge that may be relevant for answering the user's queries. This is realized by sending advertisements about the expertise of a peer. In the Oyster system, these expertise descriptions contain a set of subjects that the peer is an expert in. Peers may accept i.e. remember these advertisements, thus creating a semantic link to the other peer. These semantic links form a semantic topology, which is the basis for intelligent query routing.

Query Replier: Coordinating component controlling the process of distributing queries. It receives queries from the user interface or from other peers. Either way it tries to answer the query or distribute it further according to the content of the query. The decision to which peers a query should be sent is based on the knowledge about the expertise of other peers.

User Interface: The user interface allows the user to import, create and edit ontology metadata as well as to easily formulate queries.

8.4 Implementation details

Oyster system is freely available for download at Oyster home site `http://oyster.ontoware.org/`. Besides the latest releases of Oyster (Windows version and linux

[2] `http://www.sun.com/software/jxta/`
[3] `http://www.openrdf.org/`
[4] `http://www.openrdf.org/doc/sesame/users/ch06.html`

version), the site contains a description of the system, technical information, usage information, snapshots of the system and related publications.

8.4.1 Installation of Oyster

Currently, Oyster is supported in two operating systems, i.e. Windows and linux. We are working on the Macintosh release.

Windows version: Download the latest release for windows (oyster-win32-installxxx.jar), execute the file and proceed as follow:

- Click next on the welcome window
- Select *Autostart Oyster on system startup* check box if you want to start Oyster automatically every time you start windows
- select the installation path
- click next when the *Overall installation progress* bar completes
- Select a program group for the shortcuts and the name for the new program shortcut
- finish the process clicking on done

To uninstall Oyster, execute the file *uninstaller.jar* that was generated in the directory *programfiles/Oyster application/unistaller*

Linux version: Download the latest release for linux (Oyster-xxx-linuxx.zip), unzip the file within the directory you want to install Oyster and run the script *start.sh*.

To uninstall Oyster, just delete the directory where Oyster is installed.

8.4.2 Running Oyster

After the installation of Oyster, the first time you run the system, the jxta configuration window will ask for the name of the Peer. The name of the Peer should be at least 4 characters long. The name is not necessary to be unique in the network, since internally the system uses jxta id, however is recommended that the name reflects some useful information i.e. name of the user+organisation.

8.4.3 Using Oyster

Querying

Scope: You can select the scope of your query:

- Local Peer: Restrict your query to your local peer
- Automatic Search: Intelligent selection of peers
- Selected Peers: Select a set of peers

Search details: You can restrict your query to special attributes (e.g. search for ontologies in OWL), or for ontologies about a specific subject (based on the classification ontology loaded, such as DMOZ), or you can search for a string match in any attribute. Furthermore, you can search ontologies by their namespace (uri), and the result will show all the ontologies that includes the words on the uri field on their namespace.

Save items

You can save items to your local node repository by simply selecting the item and pressing the ”Save” button.

Adjusting the columns

By pressing the right mouse button inside the result view panel you can adjust your personal view.

Create and Edit items

New ontology metadata entries can be created from scratch, or existing ones can be modified. Oyster provides two templates for default to create a new metadata entry from scratch: OntologyDocumentFull which includes all the attributes proposed in OMV, and OntologyDocumentRequired which includes only the attributes the OMV propose as required. You can also create your own templates or you can create the metadata entry without using templates and just adding the attributes you want to include. It is important to notice that the docName and uri attributes are mandatory in order to create a new metadata entry.

Import functions

It is possible to import an ontology file document to extract the ontology metadata and then fill missing values. Oyster supports the OWL, DAML+OIL and RDF-S ontology languages. Also, a complete repository (a RDF file) can be imported into the local repository.

Export repository

The repository can be exported to a HTML file (based on a configurable XSLT file) or to a RDF file. It is also possible to export individual entries.

Preferences

The classification ontology loaded by default is based on DMOZ, but it is possible to load different classification ontologies by changing which file to open in the preferences menu. You can also change the location of the local repository, the templates, or OMV ontology.

Grouping

The aggregate function, group the Ontology Implementations that are different realizations of the same Ontology as proposed by OMV.

8.5 Preliminary evaluation

So far, Oyster system has been tested by UPM with the collaboration of UKarl (approx 10 peers), however all the Knowledge Web partners were informed of the availability of Oyster system for their evaluation. The main results of the evaluations were:

- Include the Namespace for searching ontologies based on that property,
- Include a description of the use of templates, which was include in the site,
- Use user friendly name in the user interface for the properties of the metadata,
- Include hints for the meaning of each property in the user interface,
- Include a link for a more detailed description of the properties in the help menu,
- Minor fixes in the interface,
- Include a Macintosh version of Oyster (on progress)

The latest release of Oyster has been downloaded 76 times (69 windows + 7 linux) from ontology engineering platform OntoWare[5]. It is ranked as the number one in the list of top downloaded projects of Ontoware (570 downloads, including all versions and releases). Currently we are working to include statistics every month about number of peers, downloads, number of ontologies, etc. as well as use inside and outside Knowledge Web.

[5] `http://ontoware.org/`

Chapter 9

ONTHOLOGY — Ontology Metadata Portal

As the importance of metadata increases with the number of existing ontologies, the demand for a supporting technologies like storage and access techniques becomes important as well. We present the conceptual design of a centralised ontology metadata portal and its implementation, so-called ONTHOLOGY standing for “anthology of ontologies”.

9.1 Scope

Centralised systems allow to reflect long-term community processes in which some ontologies become well accepted for a domain or community and others become less important. Such well accepted ontologies and in particular their metadata need to be stored in a central metadata portal which can be accessed easily by a large number of users whereby the management procedures are well defined. Hence, a main goal of a centralised metadata portal is to act as large evidence storage of metadata resp. their related ontologies to facilitate access, reuse and sharing as required for the Semantic Web.

9.2 Actors

We identified several different user roles for ONTHOLOGY : The *visitor* is an anonymous user, he is allowed to browse the public content of the portal. A Visitor can become a *user* by completing an application form on the website. In order to avoid unnecessary administrative work, a user is added automatically to the membership database. Users can customize their portal, e.g. the content of their start-page or their bookmarks. If a user wants to submit metadata to the portal, this submission has to be reviewed before it is published. ONTHOLOGY establishs a *review process* in order to ensure a certain level

of quality. *Reviewers* check the new submissions before it is published. The *technical administrator* is responsible for any other task mainly the maintenance of the portal.

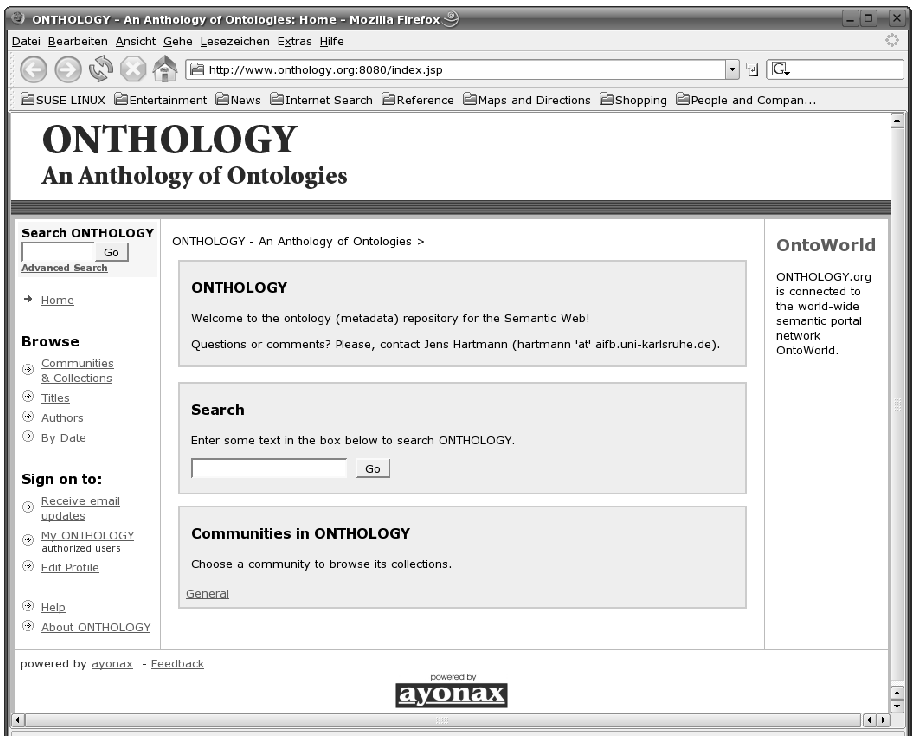

Figure 9.1: The ONTHOLOGY portal

9.3 Functionalities

Functionalities of ONTHOLOGY can be separated into two groups based on the usage. Indeed, *basic functionalities* which are provided to every user who accesses the portal and *sophisticated functionalities* for reviewers and administrators. The main operations a user can perform on the repository are (i) *Search*, (ii) *Submit* and (iii) *Export*, c.f. figure 9.1.

The search and export can be performed by any visitor without being registered to the repository. Since providing new metadata is based on a certain community confidence, a visitor has to register at the portal to be become a registered user.

9.4 Architecture

A metadata portal mainly consists of *a large data repository* in which metadata can be stored. Exemplary, Sesame[1] or KAON[2] can be used as back-end metadata repository. Furthermore, *access* and in particular the *managament* of the repository must be guaranteed, too. Therefore, ONTHOLOGY is based on SEAL, the AIFB conceptual architecture for building SEmantic portALs. In SEAL ontologies are key elements for managing community web sites and web portals. They support queries to multiple sources, but beyond that also intensive use of the schema information itself to allow for automatic generation of navigational views such as navigation hierarchies that appear as `has-part`-trees or `has-subtopic` trees in the ontology. In addition to that mixed ontology and content-based presentation is supported. Further information can be found at [HS04].

9.5 Implementation Details

The ONTHOLOGY project uses the collaborative software and ontology engineering plattform OntoWare[3] for development and dissemination.

The project can be accessed at `http://ontoware.org/projects/onthology` where the actual software version, news, examples and reference applications can be found as well.

ONTHOLOGY follows the vision of Open Source (OS) software and therefore applies existing OS projects and provides its own code as OS.

Technically, the current version of ONTHOLOGY is based on DSpace[4] and has been extended to fit the requirements of an ontology metadata portal. For installation, we refer to the standard installation procedure of DSpace which requires mainly a Java Tomcat server.

[1] `http://www.openrdf.org/`
[2] `http://kaon.semanticweb.org/`
[3] see http://www.OntoWare.org for details
[4] c.f. http://www.dspace.org

Chapter 10

Related Work

Metadata and metadata standards have a long-tradition in a variety of computer sciences areas, such as digital libraries or data management and maintenance systems. We will briefly mention related metadata standards, including in particular those ones relevant for the Semantic Web. The **Dublin Core (DC)** metadata standard is a simple yet effective element set for describing a wide range of networked resources[1]. It includes two levels: Simple (with fifteen elements) and Qualified, including an additional element as well as a group of element refinements (or qualifiers) that adapt the semantics of the elements for resource discovery purposes. The **Reference Ontology**[AGPLTP00] is a domain ontology that gathers, describes and has links to existing ontologies. However its focus is to characterize ontologies from the user point of view, and provides only a list of property-value pairs for describing ontologies. The Semantic Web search engine **SWOOGLE**[D$^+$04] makes use of an implicitly defined metadata schema, which covers information which can be extracted automatically from ontology implementations. Our approach includes and extends this metadata vocabulary. Ideally, future versions of SWOOGLE would also take into account the additional vocabulary defined in OMV . Further on, the issue of creating a metadata standard for ontologies is addressed by various ontology repositories initiatives. However, the majority of these repositories rely on a restricted, implicitly declared vocabulary, whose meaning is not machine-understandable. The **DAML ontology library** provides a catalog of DAML ontologies that can be browsed by different properties[2]. The **FIPA ontology service**[S$^+$01] defines an agent wrapper of open knowledge base connectivity. The **SchemaWeb Directory** is a repository for RDF schemas expressed in RDFS, OWL and DAML+OIL[3]. Finally we mention the ontology metadata example presented in [Kno04] emerged within the EU Network of Excellence **Knowledge Web**. The metadata consists only of attribute-value pairs, and does not consider the distinction between conceptualizations and implementations, thus lacking of the benefits provided by OMV i.e. an efficient mechanism for

[1] c.f. `http://dublincore.org/`
[2] c.f. `http://www.daml.org/ontologies/`
[3] c.f. `http://www.schemaweb.info`

tracking several versions and evolvements of ontologies as well as for different representations of one knowledge model in different languages. However, the work presented there provided a preliminary basis for the OMV ontology introduced in this deliverable.

Chapter 11

Conclusion

To conclude, reusing existing ontologies is a key issue for sharing knowledge on the Semantic Web. Our contribution aims at facilitating reuse of ontologies which are previously unknown for ontology developers by providing an Ontology Metadata Vocabulary (OMV) and two prototypical applications for decentralized (Oyster) and centralized (ONTHOLOGY) sharing of ontology metadata based on OMV .

Important aspects like quality measures which are applied to identify *good ontologies* are of great importance for an efficient integration and usage of Oyster and ONTHOLOGY However, the analysis and development of ontology evaluation procedures is out of scope for this deliverable. Such aspects will be analyzed in KWeb deliverable 1.2.9.

Next steps include the standardization of OMV on a wider scope by particularly including non-KnowledgeWeb parties in this process, followed by a close cooperation with tool providers for ontology engineering environments and applications providers for e.g. ontology based search engines to enhance their tools with support for OMV . The agreement and application of a standard on a global level will greatly facilitate the reuse of ontologies for all participating parties.

Bibliography

[AGPLTP00] J. Arpirez, A. Gomez-Porez, A. Lozano-Tello, and H. Pinto. Reference Ontology and (ONTO)2 Agent: The Ontology Yellow Pages. *Knowledge and Information Systems*, 2:387–412, 2000.

[BEH+03] Jeen Broekstra, Marc Ehrig, Peter Haase, Frank van Harmelen, Arjohn Kampman, Marta Sabou, Ronny Siebes, Steffen Staab, Heiner Stuckenschmidt, and Christoph Tempich. A metadata model for semantics-based peer-to-peer systems. In *Proceedings of the WWW'03 Workshop on Semantics in Peer-to-Peer and Grid Computing*, MAY 2003.

[BG04] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Rec. 10 February 2004, 2004. available at `http://www.w3.org/TR/rdf-schema/`.

[D+04] Li Ding et al. Swoogle: A search and metadata engine for the semantic web. In *In Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*, pages 58–61, November 2004.

[dam01] DAML+OIL (March 2001) reference description, 2001. W3C Note, available at `http://www.w3.org/TR/daml+oil-reference`.

[HS04] J. Hartmann and Y. Sure. An infrastructure for scalable, reliable semantic portals. *IEEE Intelligent Systems*, 19(3):58–65, May/June 2004.

[HSvH04] P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In *Proceedings of the International Conference on Semantics in a Networked World (ICNSW'04)*, Paris, June 2004.

[Kno04] KnowledgeWeb European Project. Identification of standards on metadata for ontologies (Deliverable D1.3.2 KnoweldgeWeb FP6-507482), 2004.

[S+01] H. Suguri et al. Implementation of fipa ontology service. In *Proc. of the Workshop on Ontologies in Agent Systems, 5th Int. Conf. on Autonomous Agents Montreal, Canada*, 2001.

[SWM04] M. K. Smith, C. Welty, and D. McGuinness. OWL Web Ontology Language Guide, 2004. W3C Rec. 10 February 2004, available at `http://www.w3.org/TR/owl-guide/`.

[TPSS05] C. Tempich, H. S. Pinto, Y. Sure, and S. Staab. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (diligent). In A. Gmez-Prez and J. Euzenat, editors, *2nd European Semantic Web Conference, ESWC 2005*, volume 3532 of *LNCS*, pages 241–256, Heraklion, Crete, Greece, MAY 2005. Springer.

Appendix A

OMV Core Ontology

In the following we describe OMV the first metadata standard for ontologies in detail whereby OMV is represented as an ontology as well.

A.1 OntologyConceptualisation (OC)

The class *OntologyConceptualisation* describes the *core idea* of an ontology. It is shared by all *ontology implementations* that *implements* one common *conceptualisation*. A conceptualisation consists of following entities categorized as introduced in sec. 4.2.

Overview class OntologyConceptualisation	
Name	OntologyConceptualisation
Type	`class`
Identifier	
Definition	See section 3.
OMV version	0.9
Comments	Class has been renamed from *OntologyBase* to *OntologyConceptualisation*.

Table A.1: Overview class OntologyConceptualisation

hasImplementation	
Name	hasImplementation
Type	`ObjectProperty, inverseOf{implements}`
Identifier	
Appropriation	`optional`
Category	General Information
Definition	Relation to an ontology implementation (OI) of this ontology conceptualisation (OC).
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.2: hasImplementation

conceptualisationName	
Name	conceptualisationName
Type	`DatatypeProperty`
Identifier	
Appropriation	`required`
Category	General Information
Definition	The name by which an ontology conceptualisation is formally known.
Domain	`omv:OntologyConceptualisation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.3: Property: conceptualisationName

conceptualisationAcronym	
Name	conceptualisationAcronym
Type	`DatatypeProperty`
Identifier	
Appropriation	`required`
Category	General Information
Definition	A short name by which an ontology conceptualisation is unformally known.
Domain	`omv:OntologyConceptualisation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.4: Property: conceptualisationAcronym

conceptualisationDescription	
Name	conceptualisationDescription
Type	`DatatypeProperty`
Identifier	
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about an ontology conceptualisation.
Domain	`omv:OntologyConceptualisation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.5: Property: conceptualisationDescription

conceptualisationDocumentation	
Name	`conceptualisationDocumentation`
Type	`DatatypeProperty`
Identifier	
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyConceptualisation`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.2
Comments	none

Table A.6: conceptualisationDocumentation

conceptualisationKeyword	
Name	`conceptualisationKeyword`
Type	`DatatypeProperty`
Identifier	
Appropriation	`optional`
Category	General Information
Definition	Set of free keywords related to an ontology conceptualisation.
Domain	`omv:OntologyConceptualisation`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.1
Comments	

Table A.7: Property: conceptualisationKeyword

conceptualisationLicense	
Name	conceptualisationLicense
Type	`ObjectProperty`
Identifier	
Appropriation	`optional`
Category	Availability Information
Definition	Encompasses the underlying license model.
Domain	`omv:OntologyConceptualisation`
Range	`omv:LicenseModel`
Cardinality	0:1
OMV version	0.1
Comments	Reference to a concrete LicenseModel.

Table A.8: Property: conceptualisationLicense

conceptualisationSubject	
Name	conceptualisationSubject
Type	`ObjectProperty`
Identifier	
Appropriation	`optional`
Category	Applicability Information
Definition	Specifies the domain topic of an ontology. Typically, the subject can be expressed as classification against established topic hierarchies such as the general purpose topic hierarchy DMOZ or the domain specific topic hierarchy ACM for the computer science domain. The idea is to recommend one classification scheme, but allow relating to others as well.
Domain	`omv:OntologyConceptualisation`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.8
Comments	none

Table A.9: Property: conceptualisationSubject

conceptualisationType	
Name	conceptualisationType
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Applicability Information
Definition	The nature of the content of the ontology. This type refers to the ontology and it should be one of the predefined.
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyType:OneOf`{`Generic, Upper-Level, Domain, Application, Task, Foundational, Linguistic`}
Cardinality	1:1
OMV version	0.1
Comments	See section A.3 for details.

Table A.10: Property: conceptualisationType

conceptualisationContributor	
Name	conceptualisationContributor
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Provenance Information
Definition	An entity responsible for making contributions to an instance.
Domain	`omv:OntologyConceptualisation`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.11: Property: conceptualisationContributor

conceptualisationCreator	
Name	conceptualisationCreator
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Provenance Information
Definition	The main responsible for an instance.
Domain	`omv:OntologyConceptualisation`
Range	`omv:Party`
Cardinality	1:n
OMV version	0.1
Comments	none

Table A.12: Property: conceptualisationCreator

conceptualisationReviewer	
entity Name	conceptualisationReviewer
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Provenance Information
Definition	An individual responsible for the revision of an instance.
Domain	`omv:OntologyConceptualisation`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.13: Property: conceptualisationReviewer

conceptualExtension	
entity Name	conceptualExtension
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	An OC which is a conceptual extension of another OC.
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyConceptualisation`
Cardinality	0:1
OMV version	0.9.1
Comments	none

Table A.14: Property: conceptualExtension

conceptualIntegration	
entity Name	conceptualIntegration
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	An OC which conceptually integrates other OCs.
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyConceptualisation`
Cardinality	0:n
OMV version	0.9.1
Comments	none

Table A.15: Property: conceptualIntegration

conceptualRelation	
entity Name	conceptualRelation
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	An OC which is conceptually similar to other OCs.
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyConceptualisation`
Cardinality	0:n
OMV version	0.9.1
Comments	none

Table A.16: Property: conceptualRelation

conceptualSpecialisation	
entity Name	conceptualSpecialisation
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	An OC which conceptually specialises another OC.
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyConceptualisation`
Cardinality	0:1
OMV version	0.9.1
Comments	none

Table A.17: Property: conceptualSpecialisation

conceptualGeneralisation	
entity Name	conceptualGeneralisation
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	An OC which conceptually generalises another OC.
Domain	`omv:OntologyConceptualisation`
Range	`omv:OntologyConceptualisation`
Cardinality	0:1
OMV version	0.9.1
Comments	none

Table A.18: Property: conceptualGeneralisation

A.2 OntologyImplementation (OI)

Aspects of specific realizations are covered modular (and extendable) by the class *Ontology Implementation (OI)*.

Overview class OntologyImplementation	
Name	OntologyImplementation
Type	`class`
Identifier	Used Identifier for this entity.
Definition	See section 3.
OMV version	0.1
Comments	none

Table A.19: Overview class OntologyImplementation

implements	
Name	implements
Type	`ObjectProperty`, inverseOf `hasImplementation`, A.1
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	General Information
Definition	Relation to the OC of which this OI is an implementation.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyConceptualisation`
Cardinality	0:1
OMV version	0.1
Comments	Normally every OI should have one conceptualisation. Only for practical reasons we allow a cardinality of null.

Table A.20: implements

implementationName	
Name	implementationName
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which an ontology implemenentation is formally known.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.21: Property: implementationName

implementationAcronym	
Name	implementationAcronym
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which an ontology implementation is unformally known.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.22: Property: implementationAcronym

implementationDescription	
Name	implementationDescription
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about an ontology implementation.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.23: Property: implementationDescription

implementationDocumentation	
Name	`implementationDocumentation`
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.2
Comments	none

Table A.24: implementationDocumentation

implementationKeyword	
Name	`implementationKeyword`
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	General Information
Definition	Set of keywords related to an ontology implementation. Typically this set includes names of the classes, properties, etc.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.1
Comments	Unclear whether to use fixed set of keywords or not.

Table A.25: Property: implementationKeyword

status	
Name	`implementationStatus`
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	General Information
Definition	It specifies the tracking information for the contents of the ontology. , i.e. Draft, Reviewed, Final.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.1
Comments	none

Table A.26: Property: implementationStatus

creationDate	
Name	creationDate
Type	DatatypeProperty
Identifier	Used Identifier for this entity.
Appropriation	required
Category	General Information
Definition	Is the date when the ontology implementation was initially created.
Domain	omv:OntologyImplementation
Range	xsd:date
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.27: Property: creationDate

modifiedDate	
Name	modifiedDate
Type	DatatypeProperty
Identifier	Used Identifier for this entity.
Appropriation	optional
Category	General Information
Definition	Date of the last modification made to the ontology implementation.
Domain	omv:OntologyImplementation
Range	xsd:date
Cardinality	0:1
OMV version	0.1
Comments	none

Table A.28: Property: modifiedDate

implementationContributor	
Name	implementationContributor
Type	ObjectProperty
Identifier	Used Identifier for this entity.
Appropriation	optional
Category	Provenance Information
Definition	An entity responsible for making contributions to an instance.
Domain	omv:OntologyImplementation
Range	omv:Party
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.29: Property: implementationContributor

implementationCreator	
Name	implementationCreator
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Provenance Information
Definition	The main responsible for an instance.
Domain	`omv:OntologyImplementation`
Range	`omv:Party`
Cardinality	1:n
OMV version	0.1
Comments	none

Table A.30: Property: implementationCreator

implementationReviewer	
entity Name	implementationReviewer
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Provenance Information
Definition	An individual responsible for the revision of an instance.
Domain	`omv:OntologyImplementation`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.1
Comments	none.

Table A.31: Property: implementationReviewer

implementationSubject	
Name	implementationSubject
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Applicability Information
Definition	Specifies the domain topic of an ontology. Typically, the subject can be expressed as classification against established topic hierarchies such as the general purpose topic hierarchy DMOZ or the domain specific topic hierarchy ACM for the computer science domain. The idea is to recommend one classification scheme, but allow relating to others as well.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.8
Comments	none

Table A.32: Property: implementationSubject

implementationType	
Name	implementationType
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Applicability Information
Definition	The nature of the content of the ontology. This type refers to the ontology and it should be one of the predefined.
Domain	`omv:OntologyImplementation`
Range	`omv:ontologyType:OneOf`{`Generic, Upper-Level, Domain, Application, Task, Foundational, Linguistic`}
Cardinality	1:1
OMV version	0.1
Comments	See section A.3 for details.

Table A.33: Property: implementationType

language	
Name	`language`
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Applicability Information
Definition	The language of the intellectual content of the ontology implementation, i.e. English, German, etc.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.34: Property: language

appliedBy	
Name	appliedBy
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Applicability Information
Definition	Declares where the ontology has been applied.
Domain	`omv:OntologyImplementation`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.35: Property: appliedBy

usedTool	
Name	usedTool
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Format Information
Definition	The name of the tool used to create the Ontology Document.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyEngineeringTool`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.36: Property: usedTool

usedOntologyEngineeringMethod	
Name	usedOntologyEngineeringMethod
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Format Information
Definition	The name of the method model used to create the ontology.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyEngineeringMethod`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.37: Property: usedOntologyEngineeringMethod

ontologyLanguage	
Name	ontologyLanguage
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Format Information
Definition	It specifies the ontology mark-up language. i.e. RDF(S), OIL, DAML+OIL, and OWL.
Domain	`omv:OntologyImplementation`
Range	`omv:ontologyLanguage:OneOf`{`OWL-Light, OWL DL, OWL Full, DLP, DAML+OIL,...`}
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.38: Property: ontologyLanguage

ontologySyntax	
Name	ontologySyntax
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Format Information
Definition	It specifies the presentation syntax for the ontology language. , i.e. RDF/XML.
Domain	`omv:OntologyImplementation`
Range	`omv:ontologySyntax:OneOf`{`OWL XML, RDF(S), ...`}
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.39: Property: ontologySyntax

formalityLevel	
Name	formalityLevel
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Format Information
Definition	Level of formality.
Domain	`omv:OntologyImplementation`
Range	`omv:formalityLevel:OneOf{...}`
Cardinality	0:1
OMV version	0.9.1
Comments	none

Table A.40: Property: formalityLevel

ontologyURL	
Name	ontologyURL
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Availability Information
Definition	Is the URL where the ontology implementation can be found.
Domain	`omv:OntologyImplementation`
Range	`omv:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.41: Property: ontologyURL

versionInfo	
Name	versionInfo
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Availability Information
Definition	Specifies the version information of the ontology implementation. Versioning could be useful for tracking, comparing and merging ontologies The number could be incremented by 1, or a smaller or larger value, depending on the personal preference of the author. It would be recommended to use a standard.
Domain	`omv:OntologyImplementation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.42: Property: versionInfo

implementationLicense	
Name	implementationLicense
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Availability Information
Definition	Encompasses the underlying license model.
Domain	`omv:OntologyImplementation`
Range	`omv:LicenseModel`
Cardinality	0:1
OMV version	0.1
Comments	Reference to a concrete LicenseModel.

Table A.43: Property: implementationLicense

imports	
Name	imports
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	References another ontology implementation containing definitions, whose meaning is considered to be part of the meaning of the importing ontology. Each reference consists of a URI specifying from where the ontology implementation is to be imported.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.44: Property: imports

priorVersion	
Name	priorVersion
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	Contains a reference to another ontology implementation. This identifies the specified ontology implementation as a prior version of the containing ontology implementation. It may be used to organize ontology implementations by versions.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyImplementation`
Cardinality	0:1
OMV version	0.1
Comments	Might be NULL for initial ontology.

Table A.45: Property: priorVersion

backwardCompatibleWith	
Name	backwardCompatibleWith
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	This property identifies the specified ontology implementation as a prior version of the containing ontology implementation, and further indicates that it is backward compatible with it. This also indicates that all identifiers from the previous version have the same intended interpretations in the new version.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.46: Property: backwardCompatibleWith

incompatibleWith	
Name	incompatibleWith
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Appropriation	`optional`
Category	Relationship Information
Definition	This ObjectProperty indicates that the containing ontology is a later version of the referenced ontology, but is not backward compatible with it. It can be used to explicitly state that ontology implementations cannot upgrade to use the new version without checking whether changes are required.
Domain	`omv:OntologyImplementation`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.47: Property: incompatibleWith

numClasses	
Name	numClasses
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Statistic Information
Definition	Is the number of classes in the ontology implementation.
Domain	`omv:OntologyImplementation`
Range	`xsd:unsignedLong`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.48: Property: numClasses

numProperties	
Name	numProperties
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Statistic Information
Definition	Is the number of properties in the ontology implementation.
Domain	`omv:OntologyImplementation`
Range	`xsd:unsignedLong`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.49: Property: numProperties

numIndividuals	
Name	numIndividuals
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Statistic Information
Definition	Is the number of Individuals in the ontology implementation.
Domain	`omv:OntologyImplementation`
Range	`xsd:unsignedLong`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.50: Property: numIndividuals

numAxioms	
Name	numAxioms
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	Statistic Information
Definition	Is the number of Axioms in the ontology implementation. Note that this could have different meanings depending on the ontology language.
Domain	`omv:OntologyImplementation`
Range	`xsd:unsignedLong`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.51: Property: numAxioms

A.3 OntologyType

Overview class OntologyType	
Name	OntologyType
Type	`class`
Identifier	Used Identifier for this entity.
Definition	Categorizes ontologies.
OMV version	0.3
Comments	none

Table A.52: Overview class OntologyType

typeName	
Name	typeName
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which an ontology type is formally known.
Domain	`omv:OntologyType`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.53: Property: typeName

typeAcronym	
Name	typeAcronym
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which an ontology type is unformally known.
Domain	`omv:OntologyType`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.54: Property: typeAcronym

typeDescription	
Name	typeDescription
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about an ontology type.
Domain	`omv:OntologyType`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.55: Property: typeDescription

typeDocumentation	
Name	typeDocumentation
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyType`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.2
Comments	none

Table A.56: typeDocumentation

typeDefinedBy	
Name	typeDefinedBy
Type	`ObjectProperty, inverseOf(definedOntologyType)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Provenance Information
Definition	Reference a party that defined the ontology type.
Domain	`omv:OntologyType`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.57: typeDefinedBy

A.4 Individuals

- **Generic** (van Heijst et al., 1997): represent common sense knowledge reusable across domains. These ontologies include vocabulary related to things, events, time, space, causality, behavior, function, mereology, etc. They are also known as common ontologies.
- **Upper-Level** (Gómez-Pérez et al.,2003): describe very general concepts and provide general notions under which all root terms in existing ontologies should be linked. They are also known as top-level ontologies.
- **Domain** (Mizoguchi et al., 1995): are reusable in a given specific domain (medical, pharmaceutical, engineering, law, enterprise, automobile, etc.). These ontologies provide vocabularies about concepts within a domain and their relationships, about the activities taking place in that domain, and about the theories and entityary principles governing that domain.
- **Application** (Van Heijst et al., 1997): are application-dependent. They contain all the definitions needed to model the knowledge required for a particular application. These ontologies often extend and specialize the vocabulary of the domain and of task ontologies for a given application.
- **Task** (Mizoguchi et al., 1995; Guarino, 1998): describe the vocabulary related to a generic task or activity (like diagnosing, scheduling, selling, etc.) by specializing the terms in the top-level ontologies.
- **Foundational** (Schneider, Luc): axiomatic theories about high level domain-independent categories of the real world. They constitute toolboxes of eminently reusable information modelling primitives for building application ontologies about specific domains.
- **Linguistic** (De Luca, Nürnberger): large scale lexical resources that cover most words of a language and have a hierarchical structure based on the relations between concepts. These ontologies can cover specific or general domains.

A.5 Party

Overview class Party	
Name	Party
Type	`class`
Identifier	Used Identifier for this entity.
Definition	A party is a person or organisation.
OMV version	0.4
Comments	none

Table A.58: Overview class Party

locatedAt	
Name	locatedAt
Type	`ObjectProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Availability Information
Definition	The location of a Party.
Domain	`omv:Party`
Range	`omv:Location`
Cardinality	0:n
OMV version	0.9
Comments	none

Table A.59: locatedAt

developedTool	
Name	developedTool
Type	`ObjectProperty, inverseOf(toolDeveloped)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference a tool developed by a party.
Domain	`omv:Party`
Range	`omv:OntologyEngineeringTool`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.60: developedTool

developedOntologyLanguage	
Name	developedOntologyLanguage
Type	`ObjectProperty, inverseOf(languageDeveloper)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference an ontology language developed by a party.
Domain	`omv:Party`
Range	`omv:OntologyLanguage`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.61: developedOntologyLanguage

developedOntologySyntax	
Name	developedOntologySyntax
Type	`ObjectProperty, inverseOf(syntaxDeveloper)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference an ontology syntax developed by a party.
Domain	`omv:Party`
Range	`omv:OntologySyntax`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.62: developedOntologySyntax

definedOntologyType	
Name	definedOntologyType
Type	`ObjectProperty, inverseOf(typeDefinedBy)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference an ontology type defined by a party.
Domain	`omv:Party`
Range	`omv:OntologyType`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.63: definedOntologyType

developedOntologyEngineeringMethod	
Name	developedOntologyEngineeringMethod
Type	`ObjectProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference a ontology engineering method developed by a party.
Domain	`omv:Party`
Range	`omv:OntologyEngineeringMethod`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.64: developedOntologyEngineeringMethod

specifiedLicense	
Name	specifiedLicense
Type	`ObjectProperty, inverseOf(licenseSpecifiedBy)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference a license specified by a party.
Domain	`omv:Party`
Range	`omv:LicenseModel`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.65: specifiedLicense

affiliated	
Name	affiliated
Type	`ObjectProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	References another party that is affiliated with the instance.
Domain	`omv:Party`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.2
Comments	none

Table A.66: affiliated

creatorOfConceptualisation	
Name	creatorOfConceptualisation
Type	ObjectProperty, inverseOf(conceptualisationCreator)
Identifier	Used Identifier for this element.
Appropriation	optional
Category	Relationship Information
Definition	References an ontology conceptualisation created by a party.
Domain	omv:Party
Range	omv:Conceptualisation
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.67: creatorOfConceptualisation

contributorToConceptualisation	
Name	contributorToConceptualisation
Type	ObjectProperty, inverseOf(conceptualisationContributor)
Identifier	Used Identifier for this element.
Appropriation	optional
Category	Relationship Information
Definition	References an ontology conceptualisation to which a party made contributions.
Domain	omv:Party
Range	omv:Conceptualisation
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.68: contributorToConceptualisation

reviewerOfConceptualisation	
Name	reviewerOfConceptualisation
Type	ObjectProperty, inverseOf(conceptualisationReviewer)
Identifier	Used Identifier for this element.
Appropriation	optional
Category	Relationship Information
Definition	References an ontology conceptualisation reviewed by a party.
Domain	omv:Party
Range	omv:Conceptualisation
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.69: reviewerOfConceptualisation

creatorOfImplementation	
Name	`creatorOfImplementation`
Type	`ObjectProperty, inverseOf(implementationCreator)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	References an ontology implementation created by a party.
Domain	`omv:Party`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.70: creatorOfImplementation

contributorToImplementation	
Name	`contributorToImplementation`
Type	`ObjectProperty, inverseOf(implementationContributor)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	References an ontology implementation a party made contributions.
Domain	`omv:Party`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.71: contributorToImplementation

reviewerOfImplementation	
Name	`reviewerOfImplementation`
Type	`ObjectProperty, inverseOf(implementationReviewer)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	References an ontology implementation reviewed by a party.
Domain	`omv:Party`
Range	`omv:OntologyImplementation`
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.72: reviewerOfImplementation

contributorOfReview	
Name	`contributorOfReview`
Type	`ObjectProperty, inverseOf(reviewer)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	References an ontology implementation reviewed by a party.
Domain	`omv:Party`
Range	`omv:OntologyReview`
Cardinality	0:n
OMV version	0.9
Comments	none

Table A.73: contributorOfReview

A.6 Person

Person represents an individual responsible for the creation, contribution or revision of an Ontology Base or Ontology Document.

Overview class Person	
Name	Person
Type	`class`
Identifier	Used Identifier for this entity.
Definition	A person.
OMV version	0.1
Comments	none

Table A.74: Overview class Person

lastName	
Name	lastName
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`required`
Category	General Information
Definition	The surname of a Person.
Domain	`omv:Person`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.2
Comments	none

Table A.75: lastName

firstName	
Name	firstName
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`required`
Category	General Information
Definition	The first name of a Person.
Domain	`omv:Person`
Range	`xsd:string`
Cardinality	1:n
OMV version	0.2
Comments	none

Table A.76: firstname

eMail	
Name	email
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`required`
Category	Availability Information
Definition	The email address of a Person.
Domain	`omv:Person`
Range	`xsd:string`
Cardinality	1:n
OMV version	0.1
Comments	none

Table A.77: eMail

phoneNumber	
Name	phoneNumber
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Availability Information
Definition	The phone number of a Person.
Domain	`omv:Person`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.78: Property: phoneNumber

faxNumber	
Name	faxNumber
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Availability Information
Definition	The fax number of a Person.
Domain	`omv:Person`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.1
Comments	none

Table A.79: faxNumber

affiliatedWith	
Name	affiliatedWith
Type	`ObjectProperty, inverseOf(employs)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Person affiliated with an organisation.
Domain	`omv:Person`
Range	`omv:Organisation`
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.80: affiliatedWith

isContactPerson	
Name	isContactPerson
Type	`ObjectProperty, inverseOf(contactPerson)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Instance is contact person of an organisation.
Domain	`omv:Person`
Range	`omv:Organisation`
Cardinality	0:n
OMV version	0.7
Comments	none

Table A.81: isContactPerson

A.7 Organisation

Represents social institutions such as universities, companies, societies etc.

Overview class Organisation	
Name	Organisation
Type	`class, subclassOf(Party)`
Identifier	Used Identifier for this entity.
Definition	An Organisation.
OMV version	0.6
Comments	none

Table A.82: Overview class Organisation

organisationName	
Name	organisationName
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which an organisation is formally known.
Domain	`omv:Organisation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.83: Property: organisationName

organisationAcronym	
Name	organisationAcronym
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which an organisation is unformally known.
Domain	`omv:Organisation`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.84: Property: organisationAcronym

employs	
Name	`employs`
Type	`ObjectProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	A person that works in that organisation.
Domain	`omv:Organisation`
Range	`omv:Person`
Cardinality	0:n
OMV version	0.2
Comments	none

Table A.85: employs

contactPerson	
Name	`contactPerson`
Type	`ObjectProperty, inverseOf(isContactPerson)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	The contact person in that Organisation.
Domain	`omv:Organisation`
Range	`omv:Person`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.86: contactPerson

A.8 LicenseModel

Overview class LicenseModel	
Name	LicenseModel
Type	`class`
Identfier	Used Identifier for this entity.
Definition	A License Model.
OMV version	0.3
Comments	none

Table A.87: Overview class LicenseModel

licenseName	
Name	licenseName
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which a license model is formally known.
Domain	`omv:OntologyLicense`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.88: Property: licenseName

licenseAcronym	
Name	licenseAcronym
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which a license model is unformally known.
Domain	`omv:OntologyLicense`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.89: Property: licenseAcronym

licenseDescription	
Name	licenseDescription
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about a license model.
Domain	`omv:OntologyLicense`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.90: Property: licenseDescription

licenseDocumentation	
Name	`licenseDocumentation`
Type	`DatatypeProperty`
Identfier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyLicense`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.2
Comments	none

Table A.91: licenseDocumentation

licenseSpecifiedBy	
Name	licenseSpecifiedBy
Type	`ObjectProperty, inverseOf(specifiedLicense)`
Identfier	Used Identifier for this element.
Appropriation	`optional`
Category	Relationship Information
Definition	Reference a party that specified the license model.
Domain	`omv:LicenseModel`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.92: licenseSpecifiedBy

A.9 OntologyEngineeringMethod

Overview class OntologyEngineeringMethod	
Name	OntologyEngineeringMethod
Type	`class`
Identfier	Used Identifier for this entity.
Definition	A method Model.
OMV version	0.3
Comments	none

Table A.93: Overview class OntologyEngineeringMethod

methodName	
Name	methodName
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which a ontology engineering method is formally known.
Domain	`omv:OntologyEngineeringMethod`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.94: Property: methodName

methodAcronym	
Name	methodAcronym
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which a ontology engineering method is unformally known.
Domain	`omv:OntologyEngineeringMethod`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.95: Property: methodAcronym

methodDescription	
Name	methodDescription
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional text about a ontology engineering method.
Domain	`omv:OntologyEngineeringMethod`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.6
Comments	none

Table A.96: Property: methodDescription

methodDocumentation	
Name	`methodDocumentation`
Type	`DatatypeProperty`
Identfier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyEngineeringMethod`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.6
Comments	none

Table A.97: methodDocumentation

methodDeveloper	
Name	methodDeveloper
Type	`ObjectProperty, inverseOf(developedKMMethod)`
Identfier	Used Identifier for this element.
Appropriation	`optional`
Category	Provenance Information
Definition	Reference a party that developed the ontology engineering method.
Domain	`omv:OntologyEngineeringMethod`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.98: methodDeveloper

A.10 OntologyEngineeringTool

Overview class OntologyEngineeringTool	
Name	OntologyEngineeringTool
Type	`class`
Identifier	Used Identifier for this entity.
Definition	An OntologyEngineeringTool.
OMV version	0.3
Comments	none

Table A.99: Overview class OntologyEngineeringTool

toolName	
Name	toolName
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which a tool is formally known.
Domain	`omv:OntologyEngineeringTool`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.100: Property: toolName

toolAcronym	
Name	toolAcronym
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which a tool is unformally known.
Domain	`omv:OntologyEngineeringTool`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.101: Property: toolAcronym

toolDescription	
Name	toolDescription
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about a tool.
Domain	`omv:OntologyEngineeringTool`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.102: Property: toolDescription

toolDocumentation	
Name	`toolDocumentation`
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyEngineeringTool`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.2
Comments	none

Table A.103: toolDocumentation

toolDeveloper	
Name	toolDeveloper
Type	`ObjectProperty, inverseOf(developedTool)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Provenance Information
Definition	Reference a party that developed the tool.
Domain	`omv:OntologyEngineeringTool`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.4
Comments	none

Table A.104: toolDeveloper

A.11 OntologySyntax

Overview class OntologySyntax	
Name	OntologySyntax
Type	`class`
Identifier	Used Identifier for this entity.
Definition	A syntax Model.
OMV version	0.3
Comments	none

Table A.105: Overview class OntologySyntax

syntaxName	
Name	syntaxName
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which an ontology syntax is formally known.
Domain	`omv:OntologySyntax`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.106: Property: syntaxName

syntaxAcronym	
Name	syntaxAcronym
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which an ontology syntax is unformally known.
Domain	`omv:OntologySyntax`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.107: Property: syntaxAcronym

syntaxDescription	
Name	syntaxDescription
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about an ontology syntax.
Domain	`omv:OntologySyntax`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.6
Comments	none

Table A.108: Property: syntaxDescription

syntaxDocumentation	
Name	`syntaxDocumentation`
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologySyntax`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.6
Comments	none

Table A.109: syntaxDocumentation

syntaxDeveloper	
Name	syntaxDeveloper
Type	`ObjectProperty, inverseOf(developedOntologySyntax)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Provenance Information
Definition	Reference a party that developed the ontology syntax.
Domain	`omv:OntologySyntax`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.110: syntaxDeveloper

A.12 OntologyLanguage

Overview class OntologyLanguage	
Name	OntologyLanguage
Type	`class`
Identifier	Used Identifier for this entity.
Definition	A language Model.
OMV version	0.3
Comments	none

Table A.111: Overview class OntologyLanguage

languageName	
Name	languageName
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which an ontology language is formally known.
Domain	`omv:OntologyLanguage`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.112: Property: languageName

languageAcronym	
Name	languageAcronym
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which an ontology language is unformally known.
Domain	`omv:OntologyLanguage`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.1
Comments	none

Table A.113: Property: languageAcronym

languageDescription	
Name	languageDescription
Type	`DatatypeProperty`
Identifier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about an ontology language.
Domain	`omv:OntologyLanguage`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.6
Comments	none

Table A.114: Property: languageDescription

languageDocumentation	
Name	`languageDocumentation`
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyLanguage`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.6
Comments	none

Table A.115: languageDocumentation

languageDeveloper	
Name	languageDeveloper
Type	`ObjectProperty, inverseOf(developedOntologyLanguage)`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	Provenance Information
Definition	Reference a party that developed the ontology language.
Domain	`omv:OntologyLanguage`
Range	`omv:Party`
Cardinality	0:n
OMV version	0.6
Comments	none

Table A.116: languageDeveloper

A.13 OntologyTask

Overview class OntologyTask	
Name	OntologyTask
Type	`class`
Identfier	Used Identifier for this entity.
Definition	A task Model.
OMV version	0.9.1
Comments	none

Table A.117: Overview class OntologyTask

taskName	
Name	taskName
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	The name by which an ontology task is formally known.
Domain	`omv:OntologyTask`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.9.1
Comments	none

Table A.118: Property: taskName

taskAcronym	
Name	taskAcronym
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	A short name by which an ontology task is unformally known.
Domain	`omv:OntologyTask`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.9.1
Comments	none

Table A.119: Property: taskAcronym

taskDescription	
Name	taskDescription
Type	`DatatypeProperty`
Identfier	Used Identifier for this entity.
Appropriation	`required`
Category	General Information
Definition	Descriptional free text about an ontology task.
Domain	`omv:OntologyTask`
Range	`xsd:string`
Cardinality	1:1
OMV version	0.9.1
Comments	none

Table A.120: Property: taskDescription

taskDocumentation	
Name	`taskDocumentation`
Type	`DatatypeProperty`
Identfier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Is a URL for further documentation.
Domain	`omv:OntologyTask`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.9.1
Comments	none

Table A.121: taskDocumentation

A.14 Location

The geographical location of a party. To keep things simple we use only DatatypeProperties instead of introducing classes for land etc.

Overview class Location	
Name	Location
Type	`class`
Identifier	Used Identifier for this entity.
Definition	A location.
OMV version	0.9
Comments	none

Table A.122: Overview class Location

state	
Name	state
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	The state of a land.
Domain	`omv:Location`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.9
Comments	none

Table A.123: state

land	
Name	land
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Name of land.
Domain	`omv:Location`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.9
Comments	none

Table A.124: land

city	
Name	city
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Name of city (and zip code).
Domain	`omv:Location`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.9
Comments	none

Table A.125: city

street	
Name	street
Type	`DatatypeProperty`
Identifier	Used Identifier for this element.
Appropriation	`optional`
Category	General Information
Definition	Name of street and number (adress).
Domain	`omv:Location`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.9
Comments	none

Table A.126: street