O M V
Ontology Metadata Vocabulary
for the Semantic Web

Raúl Palma (Universidad Politecnica de Madrid),
Jens Hartmann (University of Bremen),
Elena Paslaru Bontas (Free University of Berlin) and
Peter Haase (Universität Karlsruhe (TH))

Document Identifier	OMV Report
Project	Ontology Metadata Vocabulary
Version	v2.3
Date	September, 2007
State	Final
Distribution	public

OMV Consortium

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

University of Bremen (TZI)
TZI - Center for Computing Technologies
Universität Bremen
D-28359 Bremen
Germany
Fax: +49 421 2182449, Phone: +49 421 2187196
Contact person: Jens Hartmann
E-mail address: jh@tzi.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

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

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe (TH)
D-76128 Karlsruhe
Germany
Fax: +49 721 6086580, Phone: +49 721 6089705
Contact person: Peter Haase
E-mail address: haase@aifb.uni-karlsruhe.de

Changes

Version 2.3

- dataProperty Added: containsTBox.
- dataProperty Added: containsRBox.
- dataProperty Added: containsABox
- dataProperty Added: expressiveness
- objectProperty Added: endorsedBy
- Renaming: consistencyAccordingToReasoner - isConsistentAccordingToReasoner
- name cardinality set to >=1
- resourceLocator cardinality set to =1
- naturalLanguage reference to ISO 639
- Section Updated: identification, versioning and location
- minor fixes

Version 2.2

- Section Added: identification, versioning and location
- dataProperty Added: consistencyAccordingToReasoner.
- dataProperty Added: keyClasses.
- dataProperty Added: notes
- objectProperty Added: knownUsage
- Pre-defined instances updated for formality level class.

Version 2

- Moved Ontology Conceptualisation class into an Extension.
- Definition of the naming convention used.
- Renaming: Ontology Implementation class - Ontology
- Renaming: Ontology Conceptualisation class - Conceptualisation
- Several re-namings of properties.
- New class OntologyDomain

Version 0.9.5

- Strict naming conventions applied for OC and OI. Hence several re-namings.
- Moved *Reviewer attributes into Evaluation extension
- Changed several cardinalities
- ReNaming: OI.language - naturalLangugage
- ReNaming: OI.usedTool - usedOntologyEngineeringTool
- ReNaming: OI.ontologyLanguage - representedByOntologyLanguage
- ReNaming: OI.ontologySyntax - representedByOntologySyntax
- ReNaming: OI.versionInfo - implementationVersion
- ReNaming: OI.formalityLevel - implementationFormalityLevel
- New Class *OntologyFormalityLevel*
- ReNaming: OI.ontologyURL - implementationURL
- ReNaming: OI.imports - implementationImports
- ReNaming: OI.priorVersion - implementationPriorVersion
- ReNaming: OI.backwardCompatibleWidth - imeplementationBackwardCompatibleWith
- ReNaming: OI.incompatibleWith - implementationIncompatibleWith
- ReNaming: OI.num* - implementationNum*
- ReNaming: Party.developedTool - developedEngineeringTool
- ReNaming: Party.specifiedLicense - specifiedLicenseModel
- ReNaming: Person.* - Person.person*
- ReNaming: Organisation.* - Organisation.organisation*
- ReNaming: Organisation.contactPerson - hasContactPerson

Version 0.9.1

- Rename of *ontology document* into *OntologyImplementation*
- Moved class OntologyReview into Evaluation Extension
- Several re-namings
- New extensions: OntologyApplication, OntologyUsage, Directives, ...
- Introduced property formalityLevel for OntologyImplementation
- New class OntologyTask

Version 0.9

- Rename of *ontology base* into *conceptualisation*
- Introduced class OntologyReview
- New objectproperty contributorOfReview for class party
- Several re-namings due introduction of conceptualisation
- Introduced class `Location`. Hence, removed property adress from class party.

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. authorship information, such as provided by Dublin Core for text documents. This burden makes it difficult for academia and industry e.g. to identify, find and apply – basically meaning to reuse – ontologies effectively and efficiently.

Hence we propose a metadata vocabulary for ontologies, so called Ontology Metadata Vocabulary (OMV).

Contents

Chapter 1

Introduction

Ontologies are intended to be used as a shared means of communication between computers and between humans and computers. A core requirement for the achievement of this goal is the usage of open standards and technologies for the representation, description, access and exchange of the ontological sources. Consider, for example, the W3C standardized Web Ontology Language OWL [13]. Using this representation language instead of a proprietary format would clearly increase the usability of an ontology at Web scale. The same applies for the means employed to describe existing ontologies or for the technological infrastructure supporting their management and exchange.

In contrast to plain Web documents, the majority of implemented ontologies are currently put into widespread use on the Web without any additional metadata information. This deficiency seriously affects the reusability of Semantic Web ontologies: without any metadata information potential ontology users can not find and deploy them effectively and efficiently. In order to cope with this problem, it is necessary to agree on a *standard for ontology metadata*, a vocabulary of terms and definitions describing ontologies. Replicating the positive experiences in other information management areas e.g. Digital Libraries, implementing such a vocabulary in conjunction with a solid technological infrastructure for creating, maintaining and distributing metadata is expected to increase the real value of ontologies by facilitating their wide scale sharing and reuse.

In this report we describe our contribution to the alleviation of this situation: the ontology metadata standard OMV (**O**ntology **M**etadata **V**ocabulary), which specifies reusability-enhancing ontology features for human and machine processing purposes. The remaining of this report is organized as follows: after clarifying the applied terminology and naming conventions (Chapter 2) we perform an analysis of the requirements for the realization of the proposed ontology metadata scheme in Chapter 3. We introduce the main ideas behind the OMV vocabulary and give a detailed description of the metadata and its extensions in Chapters 4, 5 and 6, respectively. The usage of the metadata is illustrated in Chapter 7. Finally we summarize our work and sketch its current limitations and future research directions in Chapter 8.

Chapter 2

Preliminary considerations

2.1 Terminology

In this section we clarify our understanding of the concept of metadata for ontologies:

- **Metadata** - data about data
- **Ontology Metadata** - metadata which provides information about ontologies
- **Metadata Ontology** - an ontology representing metadata information
- **Metadata Entity** - an element of a metadata scheme
- **OMV** - **O**ntology **M**etadata **V**ocabulary - The acronym of the proposed ontology metadata scheme
- Metadata Categories - we differentiate among the following three occurrence constraints for metadata elements, according to their impact on the prospected reusability of the described ontological content:
  - **Required** - mandatory metadata elements. Any missing entry in this category leads to an incomplete description of the ontology.
  - **Optional** - important metadata facts, but not strongly required.
  - **Extensional** - specialized metadata entities, which are not considered to be part of the core metadata scheme.

  Complementary to this classification we organize the metadata elements according to the type and purpose of the contained information as follows:

  - **General** - elements providing general information about the ontology.

- **Availability** - information about the location of the ontology (e.g. its URI or URL where the ontology is published on the Web)
- **Applicability** - information about the intended usage or scope of the ontology.
- **Format** - information about the physical representation of the resource. In terms of ontologies these elements include information about the representation language(s) in which the ontology is formalized.
- **Provenance** - information about the organizations contributing to the creation of the ontology.
- **Relationship** - information about relationships to other resources. This category include versioning, as well as conceptual relationships such as extensions, generalization/specialization and imports.
- **Statistics** - various metrics on the underlying graph topology of an ontology (e.g. number of classes)
- **Other** - information not covered in the categories listed above.

Note that the introduced classification dimensions are not intended to be part of the metadata scheme itself, but will be taken into consideration by the implementation of several metadata support facilities. The first dimension is relevant for a metadata creation service in order to ensure a minimal set of useful metadata entries for each of the described resources. The second can be used in various settings mainly to reduce the user-perceived complexity of the metadata scheme whose elements can be structured according to the corresponding classes.

2.2 Naming conventions

Choosing a naming convention for ontology modeling and adhere to these conventions makes the ontology easier to understand and helps to avoid some common modeling mistakes. For the modeling of OMV we adopted the following set of conventions for classes, properties and instances:

2.2.1 Delimiters and capitalization

- **Class Names** - Class names are capitalized. If the class name contains more than one word, we use concatenated words and capitalize each new word. I.e. ”Ontology” ”OntologySyntax”
- **Property Names** - Property names use lower case. If the property name contains more than one word, we use concatenated words where the first word is all in lower case and capitalize each subsequent new word. I.e. ”name” ”naturalLanguage” ”hasLicense”

- **Instance Names** - Instance names are capitalized. If the instance name contains more than one word, each word is separated by a blank space and capitalize each word. I.e. ”Task Ontology” ”Highly Informal”

2.2.2 Prefix conventions

OMV use prefix conventions to distinguish **DatatypeProperty** and **ObjectProperty**. Thus, the ObjectProperties start with a verb specifying how the two classes are related to each other. I.e. ”specifiedBy” ”usedOntologyEngineeringTool” ”hasOntologySyntax”.

2.2.3 Singular form

The convention adopted in OMV was to use names for classes, properties and instances in singular form. The decision was based on the fact that singular form is used more often in practice in many domains. Besides, when working with XML, for example, importing legacy XML or generating XML feeds from the ontology, it is necessary to make sure to use a singular form since this is expected convention for XML tags.

2.2.4 Additional considerations

- When a word within a name is all capitals, the next word should start in lower case. An hypothetical example: ”URLoriginal”
- Do not add strings such as ”class” or ”attribute”, and so on to the names.
- Do not concatenate the name of the class to the properties or instances, i.e. there is no ”ontologyName” ”ontologySyntaxName”
- Do not use abbreviations in the names of classes or instances, and try to avoid abbreviations on property names.

2.3 Notations

In the following we give an overview of the notations used in this report for representing the OMV metadata entities. The metadata scheme is formalized as a Semantic Web ontology in OWL (the introduced examples conform to the OWL RDF-XML syntax).

Further on OMV uses the following namespaces:

```
owl ="http://www.w3.org/2002/07/owl#"
rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

Name of the OMV metadata entity	
Name Type Identfier	(Case sensitive) name of the metadata entity. The type of ontological primitive used to represent the entity in OWL: `Class, ObjectProperty or DatatypeProperty.` Unique identifier used for this entity.
Occurrence Constraint Category	One of the following: `required`, `optional` or `extensional`. The content/purpose category the entity belongs to, as introduced above.
Definition	A short definition of the purpose, which might be elaborated in the `comments` tag.
Domain Range Cardinality	Domain of OMV entity (for OWL properties) Range of OMV entity (for OWL properties). Cardinality of OMV entity (MIN:MAX).
OMV version	OMV version, in which the entity has been introduced.
Comments	Detailed description of the entity.

Table 2.1: Template for a metadata entry

```
rdfs="http://www.w3.org/2000/01/rdf-schema#"
xsd ="http://www.w3.org/2001/XMLSchema#"
omv ="http://omv.ontoware.org/ontology#"
```

A metadata entity can be a *class* or *property (DatatypeProperty, ObjectProperty)* of the OMV ontology. Every entity is described using the template illustrated by Table 2.1.

Chapter 3

Ontology Metadata Requirements

We elaborated an inventory of requirements for the metadata model as a result of a systematic survey of the state of the art in the area of ontology reuse. Besides analytical activities, we conducted extensive literature research, which focused on theoretical methods [11, 2, 7], but also on case studies on reusing existing ontologies [14, 12, 10], in order to identify the real-world needs of the community w.r.t. a descriptive metadata format for ontologies. Further on, the requirements analysis phase was complemented by a comparative study of existing (ontology-independent) metadata models and of tools such as ontology repositories and libraries (implicitly) making use of metadata-like information. Several aspects are definitely similar to other metadata standards such as Dublin Core. Differences arise however if we consider the semantic nature of ontologies, which are much more than plain Web information sources. In accordance to one of the major principles in Ontological Engineering an ontology comprises a conceptual model of a particular domain of interest, represented at knowledge level, and multiple implementations using knowledge representation languages. These two components are characterized by different properties, can be developed and maintained separately. The main requirements identified in this process step are the following:

Accessibility: Metadata should be accessible and processable for both humans and machines. While the human-driven aspects are ensured by the usage of natural language concept names, the machine-readability requirement can be implemented by the usage of Web-compatible representation languages (such as XML or Semantic Web languages, see below).

Usability: This requirement states for the necessity of building a metadata model which 1) reflects the needs of the majority of ontology users, as reported by current case studies in ontology reuse, but in the same time 2) allows proprietary extensions and refinements in particular application scenarios. From a content perspective, usability can be maximized by taking into account multiple metadata types, which correspond to specific viewpoints on the ontological resources and are applied in various application tasks. Despite the broad understanding of the metadata concept and the use cases associated to each definition, several key aspects of metadata information have already established across computer science fields [9]:

- **Structural metadata** relates to statistical measures on the graph structure underlying an ontology. In particular we mention the number of specific ontological primitives (e.g. number of classes, instances). The availability of structural metadata influences the usability of an ontology in a concrete application scenario, as size and structure parameters constraint the type of tools and methods which are applied to aid the reuse process.
- **Descriptive metadata** relates to the domain modelled in the ontology in form of keywords, topic classifications, textual descriptions of the ontology contents etc. This type of metadata plays a crucial role in the selection of appropriate reuse candidates, a process which includes requirements w.r.t. the domain of the ontologies to be re-used.
- **Administrative metadata** provides information to help manage ontologies, such as when and how it was created, rights management, file format and other technical information.

Interoperability: Similarly to the ontology it describes, metadata information should be available in a form which facilitates metadata exchange among applications. While the syntactical aspects of interoperability are covered by the usage of standard representation languages (see “Accesibility”), the semantical interoperability among machines handling ontology metadata information can be ensured by means of an formal and explicit representation of the meaning of the metadata entities, i.e. by conceptualizing the metadata vocabulary itself as an ontology.

Chapter 4

OMV - Ontology Metadata Vocabulary

This chapter gives an overview of the core design principles applied for the realization of the OMV metadata scheme, which is described in detail in the remainder of the report.

4.1 Core and Extensions

Following the usability constraints identifies during the requirements analysis, we decided to design the OMV scheme modularly; OMV distinguishes between the OMV Core and various OMV Extensions. The former captures information which is expected to be relevant to the majority of ontology reuse settings. However, in order to allow ontology developers and users to specify task- or application-specific ontology-related information we foresee the development of OMV extension modules, which are physically separated from the core scheme, while remaining compatible to its elements.

4.2 Ontological Representation

Due to the high accessibility and interoperability requirements, as well as the nature of the metadata, which is intended to describe Semantic Web ontologies, the conceptual model designed in the previous step was implemented in the OWL language. An implementation as XML-Schema or DTD was estimated to restrict the functionality of the ontology management tools using the metadata information (mainly in terms of retrieval capabilities) and to impede metadata exchange at semantical level. Further on, a language such as RDFS does not provide a means to distinguish between required and optional metadata properties. The implementation was performed manually by means of a common ontology editor.

4.3 Identification, Versioning and Location

An important issue that has to be addressed when describing ontologies is the ability to identify and manage multiple versions and physical representations of one ontology. The OWL ontology language itself does not provide the means to address this issue: In OWL, an ontology is identified by a URI. Here it is important to note, that this URI is merely a logical identifier, it does not (necessarily) relate to the physical location of the ontology (e.g. in a file), nor does it prescribe a versioning scheme.

Versioning OWL does not distinguish between the notion of an ontology and a version of an ontology at all. It may thus be that different versions of ontology carry the same logical URI.

In general, a version is a variant of an ontology that is usually created after applying changes to an existing variant. Therefore we need a way to unambiguously identify the different versions as well as to keep track of the relationships between them. Based on [5], we consider that changes in ontologies are caused by: (i) changes in the domain; (ii) changes in the shared conceptualization; (iii) changes in the specification. Taking the definition of an ontology as a specification of a conceptualization, (i) and (ii) are semantic changes that lead to the creation of a new conceptualization, while (iii) is just a change in the representation of the same conceptualization (also known as a new revision) (e.g. updates of natural language descriptions of ontology elements). In any case, the change(s) result in a different physical representation of the ontology (i.e. different version). Consequently, it should be possible to identify each of those versions. Normally an ontology is identified by an URI, which according to [1] is a compact string of characters for identifying an abstract or physical resource. In [5] the authors propose that any version that constitutes a new conceptualization (i.e. changes of type (i) and (ii)) should have a unique URI, however in practice different versions of same ontology might share the same URI. Furthermore, even if a revision constitutes the same conceptualization of an ontology it is physically represented in a different file which might have additional metadata (e.g. updated ontology description, descriptions in different natural languages, different file location, etc.).

In OMV we describes a particular representation of an ontology, i.e. an ontology in a particular version at a particular physical location. That means that every different version of an ontology has a different OMV related metadata.

Currently however, many ontologies either do not provide any version information at all or the ontology editors explicitly do not want to change the version of the ontology after making some changes. In those cases, whenever the ontology changes, the related OMV annotation will have to be updated accordingly instead of creating a new OMV instance (i.e. including updating the date of the last time the ontology was modified).

Resource Location An addition to the issue of versioning, an ontology (or a version of an ontology) can be located at different locations. Thus, ontologies with the same logical URI may exist at different physical locations, possibly even with different content.

Similar to approach for versioning, we rely on a composite identifier consisting of the logical identifier (URI plus optional version identifier) and a resource locator that specifies the actual physical location.

Of course, the optional version identifier and the optional resource locator can be combined, such that we end up with a tripartite identifier (URI, version, resource locator).

OMV identity Based on the previous discussion, we propose the following composite URI to identify an OMV instance which should be treated just as one possible approach (i.e. the system implementing OMV can choose its own OMV identity):

Ontology URI + ? [version=<version >;]location=<resourceLocator >#metadata

where the resourceLocator is the physical location of the ontology (i.e. the resource-Locator property) and version is the ontology version (i.e. the version property)

Illustrative Example In order to clarify the discussion consider the following scenario: Initially, we have the first implementation of ontology OWLODM (i.e. `http://owlodm.ontoware.org/OWL1.0`) which provides a metamodel for the ontology language OWL 1.0. A fragment of the OMV description for OWLODM version 1.0 is the following:

```
<omv:Ontology rdf:about=
"&j;OWL1.0?version=1.0;location=http://ontoware.org/frs/download.php/307/owl10.owl#metadata">
    <omv:URI rdf:datatype="&xsd;string">http://owlodm.ontoware.org/OWL1.0</omv:URI>
    <omv:version rdf:datatype="&xsd;string">1.0</omv:version>
    <omv:resourceLocator rdf:datatype="&xsd;string">
    http://ontoware.org/frs/download.php/307/owl10.owl</omv:resourceLocator>
    <omv:acronym rdf:datatype="&xsd;string">OWLODM</omv:acronym>
    <omv:description rdf:datatype="&xsd;string">OWL Object Definition Metamodel
    (ODM) allows interoperability of OWL ontologies with MOF-compatible
    software environments</omv:description>
    <omv:name rdf:datatype="&xsd;string">OWL Ontology Definition Metamodel</omv:name>
    <omv:numberOfClasses rdf:datatype="&xsd;unsignedInt">35</omv:numberOfClasses>
    <omv:numberOfProperties rdf:datatype="&xsd;unsignedInt">22</omv:numberOfProperties>
    <omv:hasCreator rdf:resource="#PeterHaase"/>
    <omv:hasDomain rdf:resource="&c;Knowledge Representation"/>
    <omv:creationDate rdf:datatype="&xsd;string">2007-02-12</omv:creationDate>
    ...
</omv:Ontology>
```

A change in the domain modelled by OWLODM (i.e. the definition of OWL 1.1) was reflected in a new *version* of the OWLODM ontology, namely version 1.1. This change leaded to a semantic change of the ontology (i.e. change of type (i)), and therefore a new URI was defined for OWLODM (i.e. `http://owlodm.ontoware.org/OWL1.1`). A fragment of the OMV description for OWLODM version 1.1 is the following:

```
<omv:Ontology rdf:about=
"&j;OWL1.1?version=1.1;location=http://ontoware.org/frs/download.php/365/owl11.owl#metadata">
    <omv:URI rdf:datatype="&xsd;string">http://owlodm.ontoware.org/OWL1.1</omv:URI>
```

```
    <omv:version rdf:datatype="&xsd;string">1.1</omv:version>
    <omv:resourceLocator rdf:datatype="&xsd;string">
    http://ontoware.org/frs/download.php/365/owl11.owl</omv:resourceLocator>
    <omv:acronym rdf:datatype="&xsd;string">OWLODM</omv:acronym>
    <omv:description rdf:datatype="&xsd;string">OWL Object Definition Metamodel
    (ODM) allows interoperability of OWL ontologies with MOF-compatible
    software environments</omv:description>
    <omv:name rdf:datatype="&xsd;string">OWL Ontology Definition Metamodel</omv:name>
    <omv:numberOfClasses rdf:datatype="&xsd;unsignedInt">76</omv:numberOfClasses>
    <omv:numberOfProperties rdf:datatype="&xsd;unsignedInt">35</omv:numberOfProperties>
    <omv:hasCreator rdf:resource="#PeterHaase"/>
    <omv:hasDomain rdf:resource="&c;Knowledge Representation"/>
    <omv:creationDate rdf:datatype="&xsd;string">2007-08-09</omv:creationDate>
    ...
</omv:Ontology>
```

Finally, a new version of OWLODM (i.e. version 1.2) was released as a result of a refinement. In this case, the change was at the level of the specification of the ontology (i.e. change type (iii)), in particular the renaming of a property and hence the URI was not updated. A fragment of the OMV description for the OWLODM version 1.2 is the following:

```
<omv:Ontology rdf:about=
"&j;OWL1.1?version=1.2;location=http://ontoware.org/frs/download.php/366/owl11.owl#metadata">
    <omv:URI rdf:datatype="&xsd;string">http://owlodm.ontoware.org/OWL1.1</omv:URI>
    <omv:version rdf:datatype="&xsd;string">1.2</omv:version>
    <omv:resourceLocator rdf:datatype="&xsd;string">
    http://ontoware.org/frs/download.php/366/owl11.owl</omv:resourceLocator>
    <omv:acronym rdf:datatype="&xsd;string">OWLODM</omv:acronym>
    <omv:description rdf:datatype="&xsd;string">OWL Object Definition Metamodel
    (ODM) allows interoperability of OWL ontologies with MOF-compatible
    software environments</omv:description>
    <omv:name rdf:datatype="&xsd;string">OWL Ontology Definition Metamodel</omv:name>
    <omv:numberOfClasses rdf:datatype="&xsd;unsignedInt">76</omv:numberOfClasses>
    <omv:numberOfProperties rdf:datatype="&xsd;unsignedInt">35</omv:numberOfProperties>
    <omv:hasCreator rdf:resource="#PeterHaase"/>
    <omv:hasDomain rdf:resource="&c;Knowledge Representation"/>
    <omv:creationDate rdf:datatype="&xsd;string">2007-08-10</omv:creationDate>
    ...
</omv:Ontology>
```

As we can see from the previous simple example, the URI is not enough to identify individually each version of the ontology. Besides, in practice not every semantic change leads to the definition of a new URI (as in this example). Even more, in this example it was enough the URI plus the version to identify each physical implementation, however it could also be possible that the same ontology version is located at two (or more) different physical location, where each of them could have even different content as we anticipated in the previous section. In that case we will also need the location of the ontology to identify a particular implementation (i.e. URI plus version plus location).

4.4 OMV core metadata entities

The main classes and properties of the OMV ontology are illustrated in Figure 4.1[1].

Additionally to the main class `Ontology` the metadata scheme contains further elements describing various aspects related to the creation, management and usage of an ontology. We will briefly discuss these in the following. In a typical ontology engineering process `Persons` or `Organisation`(s) are developing ontologies. We group these two classes under the generic class `Party` by a `subclass-of` relation. A `Party` can have several locations by referring to a `Location` individual and can *create*, *contribute* to ontological resources i.e. `Ontology Implementations`. Review details and further information can be captured in an extensional OMV module (see Chapter 6). Further on we provide information about the engineering process the ontology originally resulted from in terms of the classes `OntologyEngineeringMethodology`, `OntologyEngineeringTool` and the attributes `version`, `status`, `creationDate` and `modificationDate`. Again these can be elaborated as an extension of the core metadata scheme. The usage history of the ontology is modelled by classes such as the `OntologyTask` and `LicenceModel`. The scheme also contains a representation of the most significant intrinsic features of an ontology. Details on ontology languages are representable with the help of the classes `OntologySyntax`, `OntologyLanguage` and `KnowledgeRepresentationParadigm`. Ontologies might be categorized along a multitude of dimensions. One of the most popular classification differentiates among `application`, `domain`, `core`, `task` and `upper-level` ontologies. A further classification relies on their level of formality and types of Knowledge Representation (KR) primitives supported, introducing catalogues, glossaries, thesauri, taxonomies, frames etc. as types of ontologies. These can be modeled as instances of the class `OntologyType`, while generic formality levels are introduced with the help of the class `FormalityLevel`. The domain the ontology describes is represented by the class `OntologyDomain` referencing a pre-defined topic hierarchy such as the DMOZ hierarchy. Further content information can be provided as values of the DatatypeProperties `description`, `keywords`, and `documentation`. Finally the metadata scheme gives an overview of the graph topology of an `Ontology` with the help of several graph-related metrics represented as integer values of the DatatypeProperties `numberOfClasses`, `numberOfProperties`, `numberOfAxioms`, `numberOfIndividuals`.

We now turn to a detailed description of the OMV model and its planed extensions.

[1]Please notice, that not all classes and properties are included. The ontology is available for download in several ontology formats at `http://omv.ontoware.org/`

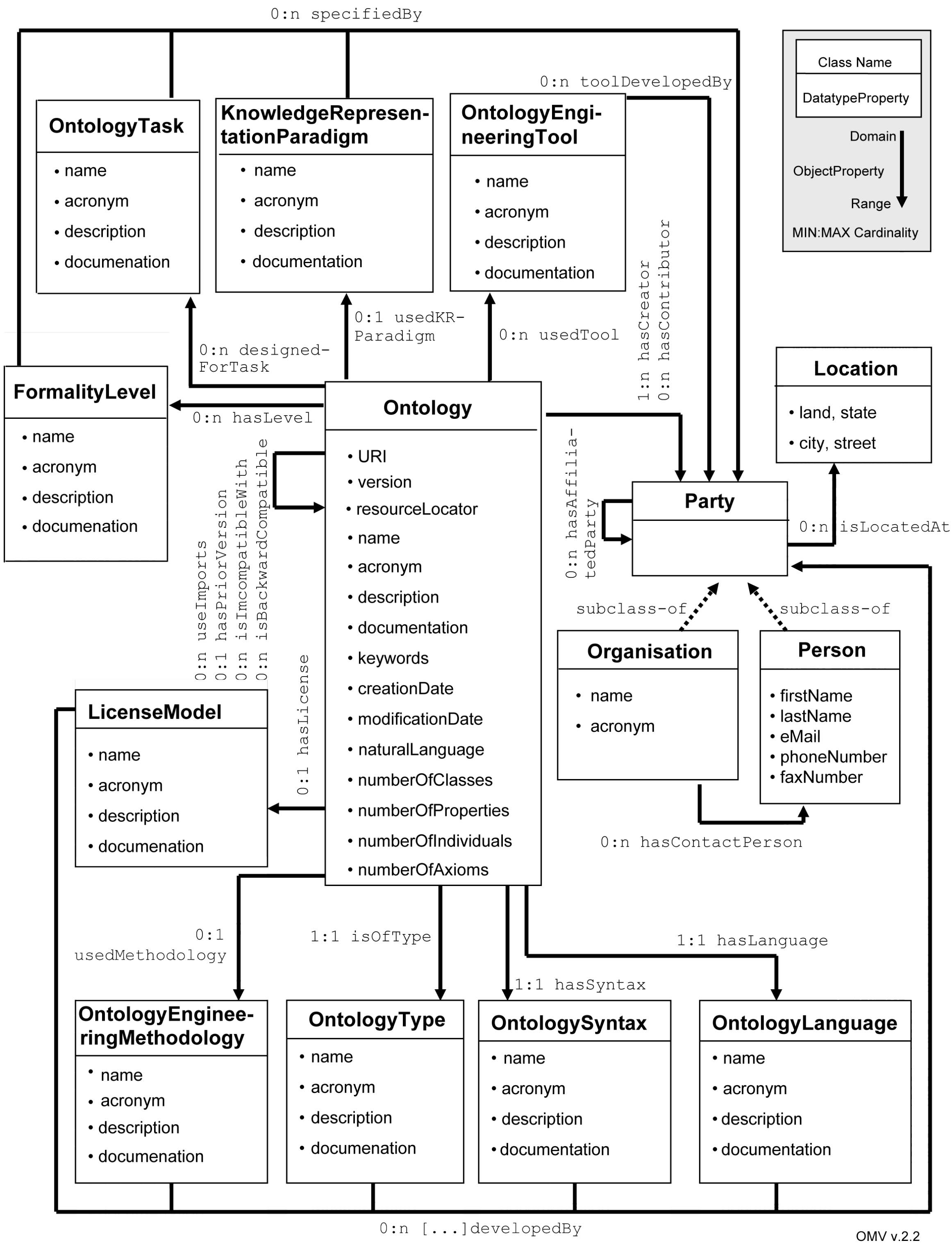

Figure 4.1: OMV overview

Chapter 5

OMV Core Ontology

In the following we introduce the metadata elements of OMV, the first metadata standard for ontologies. As aforementioned, OMV is formalized as an OWL ontology. A metadata element is modelled either by means of classes and individuals or by means of valued properties. The decision for one of these two alternatives was justified by the complexity of the corresponding metadata element. If the value/content of a metadata element can be easily mapped to conventional data types (numerical, literal, list values) the metadata element is usually represented as a DatatypeProperty. Complex metadata elements which do not fall into the previous category are modelled by means of additional classes linked by ObjectProperties.

The description of the model is grouped along the core classes of the ontology. For each class we describe the meaning of its properties and additional usage and occurrence constraints.

5.1 Ontology

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

Ontology	
Name Type Identifier	Ontology `class`
Definition	An implementation of a conceptual model
OMV version	0.1
Comments	None

Table 5.1: Class: Ontology

URI	
Name Type Identifier	URI `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	The URI of the ontology which is described by this metadata. It serves as a logical identifer and is not necessarily the physical location

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.2: Property: URI

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	The name by which an ontology is formally known

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:n

OMV version	0.1
Comments	The ontology can have many names (e.g. names in different languages)

Table 5.3: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	A short name by which an ontology is formally known

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.4: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	Free text description of an ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.5: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	0:1

OMV version	0.2
Comments	None

Table 5.6: documentation

notes	
Name Type Identifier	`notes` `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	Describes additional information about the ontology that is not included somewhere else (e.g. information that you do not want to include in the documentation)

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	0:1

OMV version	2.2
Comments	None

Table 5.7: notes

keywords	
Name	`keywords`
Type	`DatatypeProperty`
Identifier	
Occurrence Constraint	`optional`
Category	General information
Definition	List of keywords related to an ontology
Domain	`omv:Ontology`
Range	`xsd:string`
Cardinality	0:n
OMV version	0.1
Comments	Typically this set includes words that describe the content of the ontology

Table 5.8: Property: keywords

keyClasses	
Name	`keyClasses`
Type	`DatatypeProperty`
Identifier	
Occurrence Constraint	`optional`
Category	General information
Definition	Indicates what the central/best represented classes of the ontology are
Domain	`omv:Ontology`
Range	`xsd:string`
Cardinality	0:n
OMV version	2.2
Comments	none

Table 5.9: Property: keyClasses

status	
Name	`status`
Type	`DatatypeProperty`
Identifier	
Occurrence Constraint	`optional`
Category	General information
Definition	It specifies the tracking information for the contents of the ontology
Domain	`omv:Ontology`
Range	`xsd:string`
Cardinality	0:1
OMV version	0.1
Comments	Pre-defined values

Table 5.10: Property: status

creationDate	
Name	`creationDate`
Type	`DatatypeProperty`
Identifier	
Occurrence Constraint	`required`
Category	General information
Definition	Date when the ontology was initially created.
Domain	`omv:Ontology`
Range	`xsd:date`
Cardinality	1:1
OMV version	0.1
Comments	In case a versioning schema is being used, it refers to the date of creation of this particular version

Table 5.11: Property: creationDate

modificationDate	
Name Type Identifier	modifiedDate DatatypeProperty
Occurrence Constraint Category	optional General information
Definition	Date of the last modification made to the ontology.
Domain Range Cardinality	omv:Ontology xsd:date 0:1
OMV version	0.1
Comments	In case a versioning schema is being used, it is not applicable

Table 5.12: Property: modificationDate

hasContributor	
Name Type Identifier	hasContributor ObjectProperty
Occurrence Constraint Category	optional Provenance information
Definition	Contributors to the creation of the ontology
Domain Range Cardinality	omv:Ontology omv:Party 0:n
OMV version	0.1
Comments	None

Table 5.13: Property: hasContributor

hasCreator	
Name	hasCreator
Type	`ObjectProperty`
Identifier	
Occurrence Constraint	`required`
Category	Provenance information
Definition	Main responsible for the creation of the ontology
Domain	`omv:Ontology`
Range	`omv:Party`
Cardinality	1:n
OMV version	0.1
Comments	None

Table 5.14: Property: hasCreator

usedOntologyEngineeringTool	
Name	usedOntologyEngineeringTool
Type	`ObjectProperty`
Identifier	
Occurrence Constraint	`optional`
Category	Provenance information
Definition	Information about tool used to create the ontology
Domain	`omv:Ontology`
Range	`omv:OntologyEngineeringTool`
Cardinality	0:n
OMV version	0.1
Comments	None

Table 5.15: Property: usedOntologyEngineeringTool

usedOntologyEngineeringMethodology	
Name Type Identifier	usedOntologyEngineeringMethodology `ObjectProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	Information about the method model used to create the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`omv:OntologyEngineeringMethodology`
	0:n

OMV version	0.1
Comments	None

Table 5.16: Property: usedOntologyEngineeringMethodology

usedKnowledgeRepresentationParadigm	
Name Type Identifier	usedKnowledgeRepresentationParadigm `ObjectProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	Information about the paradigm model used to create the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`omv:KnowledgeRepresentationParadigm`
	0:n

OMV version	0.1
Comments	None

Table 5.17: Property: usedKnowledgeRepresentationParadigm

endorsedBy	
Name Type Identifier	endorsedBy `ObjectProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	Indicates the parties (i.e. organisations, people) that have expressed support or approval to the this ontology.

Domain
Range
Cardinality
	`omv:Ontology`
	`omv:KnowledgeRepresentationParadigm`
	0:n

OMV version	0.1
Comments	None

Table 5.18: Property: endorsedBy

hasDomain	
Name Type Identifier	hasDomain `ObjectProperty`
Occurrence Constraint Category	`optional` Applicability information
Definition	Specifies the domain topic of an ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`omv:OntologyDomain`
	0:n

OMV version	0.8
Comments	Typically, the domain can refer to established topic hierarchies such as the general purpose topic hierarchy DMOZ or the domain specific topic hierarchy ACM for the computer science domain

Table 5.19: Property: hasDomain

isOfType	
Name Type Identifier	isOfType `ObjectProperty`
Occurrence Constraint Category	`required` Applicability information
Definition	The nature of the content of the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`omv:OntologyType`
	1:1

OMV version	0.1
Comments	Pre-defined values. See section 5.2 for details

Table 5.20: Property: isOfType

naturalLanguage	
Name Type Identifier	naturalLanguage DatatypeProperty
Occurrence Constraint Category	optional Applicability information
Definition	The language of the content of the ontology, i.e. English, German, etc.
Domain Range Cardinality	omv:Ontology xsd:string 0:n
OMV version	0.1
Comments	Pre-defined values according to the names of languages defined in ISO 639

Table 5.21: Property: naturalLanguage

designedForOntologyTask	
Name Type Identifier	designedForOntologyTask ObjectProperty
Occurrence Constraint Category	optional Applicability information
Definition	Declares for which purpose the ontology was originally designed
Domain Range Cardinality	omv:Ontology omv:OntologyTask 0:n
OMV version	0.9
Comments	See Section 5.10

Table 5.22: Property: designedForOntologyTask

hasFormalityLevel	
Name Type Identifier	hasFormalityLevel ObjectProperty
Occurrence Constraint Category	optional Applicability information
Definition	Level of formality of the ontology
Domain Range Cardinality	omv:Ontology omv:FormalityLevel 0:1
OMV version	0.9.1
Comments	Pre-defined values

Table 5.23: Property: hasFormalityLevel

knownUsage	
Name Type Identifier	knownUsage DatatypeProperty
Occurrence Constraint Category	optional Applicability information
Definition	The applications where the ontology is being used
Domain Range Cardinality	omv:Ontology xsd:string 0:n
OMV version	2.2
Comments	None

Table 5.24: Property: knownUsage

hasOntologyLanguage	
Name Type Identifier	hasOntologyLanguage ObjectProperty
Occurrence Constraint Category	required Format information
Definition	Specifies the ontology language
Domain Range Cardinality	omv:Ontology omv:OntologyLanguage 1:1
OMV version	0.1
Comments	Pre-defined values

Table 5.25: Property: hasOntologyLanguage

hasOntologySyntax	
Name Type Identifier	hasOntologySyntax ObjectProperty
Occurrence Constraint Category	required Format information
Definition	It specifies the presentation syntax for the ontology language
Domain Range Cardinality	omv:Ontology omv:OntologySyntax 1:1
OMV version	0.1
Comments	Pre-defined values

Table 5.26: Property: hasOntologySyntax

isConsistentAccordingToReasoner	
Name Type Identifier	isConsistentAccordingToReasoner `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Indicates whether a reasoner has classified the ontology correctly

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:boolean`
	0:1

OMV version	2.2
Comments	The definition of consistency is independent of a reasoner. The assumption is that the reasoner used for consistency checking operates correctly, i.e. according to the well-defined semantics of the ontology language

Table 5.27: Property: isConsistentAccordingToReasoner

expressiveness	
Name Type Identifier	expressiveness `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Indicates the expressiveness of the language used by the ontology (that could be expressed in terms of the underlying DL, e.g. ALCN(D), SHIF(D), etc.)

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	0:1

OMV version	2.2
Comments	None

Table 5.28: Property: expressiveness

resourceLocator	
Name Type Identifier	resourceLocator `DatatypeProperty`
Occurrence Constraint Category	`required` Availability information
Definition	The location where the ontology can be found. It should be accessible via a URL

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:n

OMV version	0.1
Comments	None

Table 5.29: Property: resourceLocator

version	
Name Type Identifier	version `DatatypeProperty`
Occurrence Constraint Category	`required` Availability information
Definition	Specifies the version information of the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:1

OMV version	0.1
Comments	Version information could be useful for tracking, comparing and merging ontologies. It is highly recommended the use of a well defined numbering schema for the version information (e.g. X.Y.Z where X is a major release, Y is minor release and Z is a revision number)

Table 5.30: Property: version

hasLicense	
Name Type Identifier	hasLicense `ObjectProperty`
Occurrence Constraint Category	`optional` Availability information
Definition	Underlying license model

Domain
Range
Cardinality
	`omv:Ontology`
	`omv:LicenseModel`
	0:1

OMV version	0.1
Comments	Reference to a concrete LicenseModel Pre-defined values

Table 5.31: Property: hasLicense

useImports	
Name Type Identifier	useImports `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	References another ontology metadata instance that describes an ontology containing definitions, whose meaning is considered to be part of the meaning of the ontology described by this ontology metadata instance
Domain Range Cardinality	`omv:Ontology` `omv:Ontology` 0:n
OMV version	0.1
Comments	Each reference consists of a URI

Table 5.32: Property: useImports

hasPriorVersion	
Name Type Identifier	hasPriorVersion `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	Contains a reference to another ontology metadata instance
Domain Range Cardinality	`omv:Ontology` `omv:Ontology` 0:1
OMV version	0.1
Comments	Identifies the ontology metadata instance which describes an ontology that is a prior version of the ontology described by this ontology metadata instance. It may be used to organize ontologies by versions and is NULL for initial ontology

Table 5.33: Property: hasPriorVersion

isBackwardCompatibleWith	
Name Type Identifier	isBackwardCompatibleWith `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	This property identifies the ontology metadata instance which describes an ontology that is a compatible prior version of the ontology described by this ontology metadata instance
Domain Range Cardinality	`omv:Ontology` `omv:Ontology` 0:n
OMV version	0.1
Comments	This also indicates that all identifiers from the previous version have the same intended interpretations in the new version

Table 5.34: Property: isBackwardCompatibleWith

isIncompatibleWith	
Name Type Identifier	isIncompatibleWith `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	This property indicates that the described ontology is a later version of the ontology described by the metadata specified, but is not backward compatible with it. It can be used to explicitly state that ontology cannot upgrade to use the new version without checking whether changes are required.
Domain Range Cardinality	`omv:Ontology` `omv:Ontology` 0:n
OMV version	0.1
Comments	None

Table 5.35: Property: isIncompatibleWith

containsTBox	
Name Type Identifier	containsTBox `DatatypeProperty`
Occurrence Constraint Category	`optional` Statistic information
Definition	Indicates if the ontology contains assertions about concepts (e.g. subsumption, equivalence)

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:boolean`
	0:1

OMV version	2.3
Comments	). It includes concept definitions (i.e. in the form of A=C which define a concept name A by a concept description C) and general concept inclusion axioms (GCIs) (i.e. of the form C v D, where both C and D are arbitrary concept descriptions

Table 5.36: Property: containsTBox

containsRBox	
Name Type Identifier	containsRBox `DatatypeProperty`
Occurrence Constraint Category	`optional` Statistic information
Definition	Indicates if the ontology contains assertions about roles (properties) and role hierarchies (e.g. subsumption, equivalence, transitivity)

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:boolean`
	0:1

OMV version	2.3
Comments	). Among others, it includes role definitions, transitivity axioms of the form Tr(R) and role inclusion axioms: (i) of the form R v S, where R; S are roles (simple role inclusion SRI), (ii) of the form R o S v R and S o R v R where R is a role and S is a simple role (i.e. complex role inclusion)

Table 5.37: Property: containsRBox

containsABox	
Name Type Identifier	containsABox `DatatypeProperty`
Occurrence Constraint Category	`optional` Statistic information
Definition	Indicates if the ontology contains role assertions between individuals and membership assertions (e.g. concept instantiation , role instantiation)

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:boolean`
	0:1

OMV version	2.3
Comments	). It includes axioms of the form a : C, R(a, b), and a ? b, where a, b are individuals

Table 5.38: Property: containsABox

numberOfClasses	
Name Type Identifier	numberOfClasses `DatatypeProperty`
Occurrence Constraint Category	`required` Statistic information
Definition	Number of classes in the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:unsignedLong`
	1:1

OMV version	0.1
Comments	Language specific value

Table 5.39: Property: numberOfClasses

numberOfProperties	
Name Type Identifier	numberOfProperties `DatatypeProperty`
Occurrence Constraint Category	`required` Statistic information
Definition	Number of properties in the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:unsignedLong`
	1:1

OMV version	0.1
Comments	Language specific value

Table 5.40: Property: numberOfProperties

numberOfIndividuals	
Name Type Identifier	numberOfIndividuals `DatatypeProperty`
Occurrence Constraint Category	`required` Statistic information
Definition	Number of individuals in the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:unsignedLong`
	1:1

OMV version	0.1
Comments	Language specific value

Table 5.41: Property: numberOfIndividuals

numberOfAxioms	
Name Type Identifier	numberOfAxioms `DatatypeProperty`
Occurrence Constraint Category	`required` Statistic information
Definition	Number of axioms in the ontology

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:unsignedLong`
	1:1

OMV version	0.1
Comments	The meaning of axiom depends on the ontology language. For instance for a RDF(S) ontology it refers to a statement (i.e. triple) and for an OWL ontology it refers to an OWL axiom.

Table 5.42: Property: numberOfAxioms

5.2 OntologyType

This class subsumes types of ontologies according to well-known classifications in the Ontology Engineering literature [3].

OntologyType	
Name Type Identifier	OntologyType `class`
Definition	Categorizes ontologies
OMV version	0.3
Comments	None

Table 5.43: Class: OntologyType

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	The name by which an ontology type is formally known

Domain
Range
Cardinality
	`omv:OntologyType`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.44: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	A short name by which an ontology type is formally known

Domain
Range
Cardinality
	`omv:OntologyType`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.45: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	Free text description of an ontology type

Domain
Range
Cardinality
	`omv:OntologyType`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.46: Property: description

documentation	
Name Type Identifier	documentation `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:OntologyType`
	`xsd:string`
	0:1

OMV version	0.2
Comments	None

Table 5.47: documentation

definedBy	
Name Type Identifier	definedBy `ObjectProperty, inverseOf(definesOntologyType)`
Occurrence Constraint Category	`optional` General information
Definition	References a party that defined the ontology type

Domain
Range
Cardinality
	`omv:OntologyType`
	`omv:Party`
	0:n

OMV version	0.6
Comments	None

Table 5.48: typeDefinedBy

5.2.1 Pre-defined ontology types

Individuals of the class `OntologyType` refer to well-known classifications for ontologies in the literature. Currently the OMV model resorts to a classification on the generality levels of the conceptualisation [4, 15]:

- upper level ontologies describing general, domain-independent concepts e.g. space, time.
- core ontologies describing the most important concepts in a specific domain
- domain ontology describing some domain of the world
- task ontology describing generic types of tasks or activities e.g. selling, selecting.
- application ontology describing some domain in an application-dependent manner

The class can be extended to support additional classifications (e.g. the one in [8]).

5.3 LicenseModel

LicenseModel	
Name Type Identifier	LicenseModel `class` LM
Definition	A license model describing the usage conditions for an ontology
OMV version	0.3
Comments	None

Table 5.49: Class: LicenseModel

name	
Name Type Identifier	name `DatatypeProperty` Used Identifier for this entity.
Occurrence Constraint Category	`required` Availability information
Definition	The name by which a license model is formally known

Domain
Range
Cardinality
	`omv:LicenseModel`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.50: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` Availability information
Definition	A short name by which a license model is formally known

Domain
Range
Cardinality
	`omv:LicenseModel`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.51: Property: acronym

description	
Name	description
Type	DatatypeProperty
Identifier	
Occurrence Constraint	optional
Category	Availability information
Definition	Descriptional free text about a license model
Domain	omv:LicenseModel
Range	xsd:string
Cardinality	0:1
OMV version	0.1
Comments	None

Table 5.52: Property: description

documentation	
Name	documentation
Type	DatatypeProperty
Identifier	
Occurrence Constraint	optional
Category	Availability information
Definition	URL for further documentation
Domain	omv:LicenseModel
Range	xsd:string
Cardinality	0:1
OMV version	0.2
Comments	None

Table 5.53: documentation

specifiedBy	
Name	specifiedBy
Type	ObjectProperty, inverseOf(specifiesLicense)
Identifier	Used Identifier for this element.
Occurrence Constraint	optional
Category	Availability information
Definition	References a party that specified the license model
Domain	omv:LicenseModel
Range	omv:Party
Cardinality	0:n
OMV version	0.6
Comments	None

Table 5.54: specifiedBy

5.3.1 Pre-defined license models

Individuals of the class `LicenseModel` refer to well-known license models, such as: [1]

- Academic Free License (AFL)
- Common Public License (CPL)
- Lesser General Public License (LGPL)
- Open Software License (OSL)
- General Public License (GPL)
- Modified BSD License (mBSD)
- IBM Public License (IBM PL)
- Apple Public Source License (APSL)
- INTEL Open Source License (INTEL OSL)

The class can be extended to support additional classifications.

[1] A description of these models can be found in `http://www.gnu.org/licenses/license-list.html`

5.4 OntologyEngineeringMethodology

OntologyEngineeringMethodology	
Name Type Identifier	OntologyEngineeringMethodology `class`
Definition	Information about the ontology engineering methodology
OMV version	0.3
Comments	None

Table 5.55: Class: OntologyEngineeringMethodology

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` Other
Definition	The name by which a ontology engineering method is formally known

Domain
Range
Cardinality
	`omv:OntologyEngineeringMethodology`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.56: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` Other
Definition	A short name by which a ontology engineering method is known

Domain
Range
Cardinality
	`omv:OntologyEngineeringMethodology`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.57: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Other
Definition	Free text description of an ontology engineering method

Domain
Range
Cardinality
	`omv:OntologyEngineeringMethodology`
	`xsd:string`
	0:1

OMV version	0.6
Comments	None

Table 5.58: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` Other
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:OntologyEngineeringMethodology`
	`xsd:string`
	0:1

OMV version	0.6
Comments	None

Table 5.59: documentation

developedBy	

Name
Type
 	
Identifier
	developedBy
	`ObjectProperty`
	`inverseOf(developesOntologyEngineeringMethodology)`

Occurrence Constraint Category	`optional` Other
Definition	A party that developed the ontology engineering methodology

Domain
Range
Cardinality
	`omv:OntologyEngineeringMethodology`
	`omv:Party`
	0:n

OMV version	0.6
Comments	None

Table 5.60: developedBy

5.5 OntologyEngineeringTool

OntologyEngineeringTool	
Name Type Identifier	OntologyEngineeringTool `class`
Definition	A tool used to create the ontology
OMV version	0.3
Comments	None

Table 5.61: Class: OntologyEngineeringTool

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` Other
Definition	The name by which a tool is formally known

Domain
Range
Cardinality
	`omv:OntologyEngineeringTool`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.62: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` Other
Definition	A short name by which a tool is known

Domain
Range
Cardinality
	`omv:OntologyEngineeringTool`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.63: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Other
Definition	Free text description of the tool

Domain
Range
Cardinality
	`omv:OntologyEngineeringTool`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.64: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` Other
Definition	URL for further documentation.

Domain
Range
Cardinality
	`omv:OntologyEngineeringTool`
	`xsd:string`
	0:n

OMV version	0.2
Comments	None

Table 5.65: documentation

developedBy	

Name
Type
 	
Identifier
	developedBy
	`ObjectProperty`
	`inverseOf(developesOntologyEngineeringTool)`

Occurrence Constraint Category	`optional` Other
Definition	References the tool developer party

Domain
Range
Cardinality
	`omv:OntologyEngineeringTool`
	`omv:Party`
	0:n

OMV version	0.4
Comments	None

Table 5.66: developedBy

5.6 OntologySyntax

OntologySyntax	
Name Type Identifier	OntologySyntax `class`
Definition	Information about the syntax used in an OI
OMV version	0.3
Comments	None

Table 5.67: Class: OntologySyntax

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` Format information
Definition	The name by which an ontology syntax is formally known

Domain
Range
Cardinality
	`omv:OntologySyntax`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.68: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	A short name by which an ontology syntax is known

Domain
Range
Cardinality
	`omv:OntologySyntax`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.69: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Free text description of the used syntax

Domain
Range
Cardinality
	`omv:OntologySyntax`
	`xsd:string`
	0:1

OMV version	0.6
Comments	None

Table 5.70: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	URL for further documentation.

Domain
Range
Cardinality
	`omv:OntologySyntax`
	`xsd:string`
	0:1

OMV version	0.6
Comments	None

Table 5.71: documentation

developedBy	

Name
Type
 	
Identifier
	developedBy
	`ObjectProperty`
	`inverseOf(developedOntologySyntax)`

Occurrence Constraint Category	`optional` Format information
Definition	The party who developed the used syntax

Domain
Range
Cardinality
	`omv:OntologySyntax`
	`omv:Party`
	0:n

OMV version	0.6
Comments	None

Table 5.72: developedBy

5.6.1 Pre-defined ontology syntaxes

Individuals of the class `OntologySyntax` refers to well-known ontology syntax standards, such as:

- OWL-XML
- RDF/XML

The class can be extended to support additional classifications.

5.7 OntologyLanguage

OntologyLanguage	
Name Type Identifier	OntologyLanguage `class`
Definition	Information about the language in which the ontology is implemented
OMV version	0.3
Comments	None

Table 5.73: Class: OntologyLanguage

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` Format information
Definition	The name by which an ontology language is formally known

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.74: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	A short name by which an ontology language is known

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`xsd:string`
	0:1

OMV version	0.1
Comments	None

Table 5.75: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Free text description of an ontology language.

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`xsd:string`
	0:1

OMV version	0.6
Comments	None

Table 5.76: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`xsd:string`
	0:1

OMV version	0.6
Comments	None

Table 5.77: Property: documentation

developedBy	

Name
Type
 	
Identifier
	developedBy
	`ObjectProperty,`
	`inverseOf(developesOntologyLanguage)`

Occurrence Constraint Category	`optional` Format information
Definition	References the party who developed the language

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`omv:Party`
	0:n

OMV version	0.6
Comments	None

Table 5.78: Property: developedBy

supportsRepresentationParadigm	
Name Type Identifier	supportsRepresentationParadigm `ObjectProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Specifies the representation paradigm supported by the ontology language

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`omv:Party`
	0:n

OMV version	0.6
Comments	None

Table 5.79: Property: supportsRepresentationParadigm

hasSyntax	
Name Type Identifier	hasSyntax `ObjectProperty`
Occurrence Constraint Category	`optional` Format information
Definition	References the syntactical alternatives of the language

Domain
Range
Cardinality
	`omv:OntologyLanguage`
	`omv:OntologySyntax`
	0:n

OMV version	0.6
Comments	None

Table 5.80: Property: hasSyntax

5.7.1 Pre-defined ontology languages

Individuals of the class `OntologyLanguage` refer to well-known ontology language standards, such as:

- OWL
- OWL-DL
- OWL-Lite
- OWL-Full
- DAML-OIL
- RDF(S)

The class can be extended to support additional classifications.

5.8 KnowledgeRepresentationParadigm

KnowledgeRepresentationParadigm	
Name Type Identifier	KnowledgeRepresentationParadigm class
Definition	Information about a knowledge representation paradigm a particular language adheres to
OMV version	0.9.1
Comments	E. g. Description Logics, Frames

Table 5.81: Class: KnowledgeRepresentationParadigm

name	
Name Type Identifier	name DatatypeProperty
Occurrence Constraint Category	required Format information
Definition	The name by which a KR paradigm is formally known
Domain Range Cardinality	omv:KnowledgeRepresentationParadigm xsd:string 1:1
OMV version	0.9.1
Comments	None

Table 5.82: Property: name

acronym	
Name Type Identifier	acronym DatatypeProperty
Occurrence Constraint Category	optional Format information
Definition	A short name by which a kR paradigm is known
Domain Range Cardinality	omv:KnowledgeRepresentationParadigm xsd:string 0:1
OMV version	0.9.1
Comments	None

Table 5.83: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Free text description of the knowledge representation paradigm

Domain
Range
Cardinality
	`omv:KnowledgeRepresentationParadigm`
	`xsd:string`
	0:1

OMV version	0.9.1
Comments	None

Table 5.84: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:KnowledgeRepresentationParadigm`
	`xsd:string`
	0:1

OMV version	0.9.1
Comments	None

Table 5.85: documentation

specifiedBy	

Name
Type
 	
Identifier
	specifiedBy
	`ObjectProperty`
	`inverseOf specifiesKnowledgeRepresentationParadigm`

Occurrence Constraint Category	`optional` Format information
Definition	Author of the KR paradigm

Domain
Range
Cardinality
	`omv:KnowledgeRepresentationParadigm`
	`omv:Party`
	0:n

OMV version	0.1
Comments	None

Table 5.86: Property: specifiedBy

5.8.1 Pre-defined knowledge representation paradigms

In this version we foresee two main classes of `KnowledgeRepresentationParadigm`s:

- Description Logics
- Frames

5.9 FormalityLevel

FormalityLevel	
Name Type Identifier	FormalityLevel `class`
Definition	The level of formality of an ontology
OMV version	0.9.1
Comments	According to classifications in the OE literature

Table 5.87: Class: FormalityLevel

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` Applicability information
Definition	The name by which this element is formally known

Domain
Range
Cardinality
	`omv:FormalityLevel`
	`xsd:string`
	1:1

OMV version	0.9.1
Comments	None

Table 5.88: Property: name

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Format information
Definition	Free text description of the formality level

Domain
Range
Cardinality
	`omv:FormalityLevel`
	`xsd:string`
	0:1

OMV version	0.9.1
Comments	None

Table 5.89: Property: description

5.9.1 Pre-defined formality levels

The pre-defined values for the formality level are based on the work presented in [6], which classifies ontologies in a spectrum of definitions according to the detail in their specification as: catalog, glossary, thesauri, taxonomy, frames and properties, value restrictions, disjointness, general logic constraints.

5.10 OntologyTask

OntologyTask	
Name Type Identifier	OntologyTask `class`
Definition	Information about the task the ontology was intended to be used for
OMV version	0.9.1
Comments	Super-class of classes modelling typical ontology-related tasks

Table 5.90: Class: OntologyTask

name	
Name Type Identifier	taskName `DatatypeProperty`
Occurrence Constraint Category	`required` Applicability information
Definition	The name by which an ontology task is formally known

Domain
Range
Cardinality
	`omv:OntologyTask`
	`xsd:string`
	1:1

OMV version	0.9.1
Comments	None

Table 5.91: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`optional` Applicability information
Definition	A short name by which an ontology task is known

Domain
Range
Cardinality
	`omv:OntologyTask`
	`xsd:string`
	0:1

OMV version	0.9.1
Comments	None

Table 5.92: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`optional` Applicability information
Definition	Free text description of the ontology task

Domain
Range
Cardinality
	`omv:OntologyTask`
	`xsd:string`
	0:1

OMV version	0.9.1
Comments	None

Table 5.93: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` Applicability information
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:OntologyTask`
	`xsd:string`
	0:1

OMV version	0.9.1
Comments	None

Table 5.94: documentation

5.10.1 Pre-defined ontology tasks

Individuals of the class `OntologyTask` refer to particular application scenarios for ontologies, in which the benefits of using ontologies are widely acknowledged. We differentiate among the following tasks:

AnnotationTask : the ontology is used as a controlled vocabulary to annotate Semantic Web resources. This task includes the usage of a semantically rich ontology for representing arbitrarily complex annotation statements on these resources. The task can be performed manually or (semi-)automatically.

ConfigurationTask : the ontology is designed to provide a controlled and unambiguous means to represent valid configuration profiles in application systems. As the aim of the ontology is to support the operationalization of particular system-related processes; this task is performed automatically in that the ontology is processed in an automatic manner by means of reasoners or APIs.

FilteringTask : the task describes at a very general level how ontologies are applied to refine the solution space of a certain problem, such as information retrieval or personalization. The task is targeted at being performed semi-automatically or automatically.

IndexingTask : in this scenario, the goal of the ontology is to provide a clearly defined classification and browsing structure for the information items in a repository. Again, the task can be performed manually by domain experts or as part of an application in an automatic or semi-automatic way.

IntegrationTask : the task characterizes how ontologies provide an integrating environment, an inter-lingua, for information repositories or software tools. In this scenario the ontology is applied (semi-)automatically to merge between heterogeneous data pools in the same or in adjacent domains.

MatchingTask : the goal of matching is to establish links between semantically similar data items in information repositories. In contrast to the previous task, matching does not include the production of a shared final schema/ontology as a result of aggregating the matched source elements to common elements. W.r.t. the automatization level the range varies from manual to fully-automatical execution.

MediationTask : the ontology is built to reduce the ambiguities between communicating human or machine agents. It can act as a normative model which formally and clearly defines the meaning of the terms employed in agent interactions. In the context of programmed agents, the task is envisioned to be performed automatically.

QueryFormulationTask : the ontology is used in information retrieval settings as a controlled vocabulary for representing user queries. Usually the task is performed automatically in that the concepts of the ontology is are listed in a query formulation front-end in order to allow users to specifies their queries.

QueryRewritingTask : complementary to the query formulation dimension, this task applies ontologies to semantically optimize query expressions by means of the domain knowledge (constraints, subsumption relations etc.) The task can be interpreted as a particular art of filtering information. The task is performed automatically; however, it assumes the availability of patterns describing the transformations at query level.

PersonalizationTask : the ontology is used mainly for providing personalized access to information resources. Individual user preferences w.r.t. particular application settings are formally specified by means of an ontology, which, in conjunction with appropriate reasoning services, can be directly integrated to a personalization component for filtering purposes. The usage of ontologies in personalization tasks might be carried out in various forms, from a direct involvement of the user who manually specifies ontological concepts which optimally describe his preferences, to the ontological modelling of user profiles.

SearchTask : the task characterizes how ontologies are used to refine common keyword-based search algorithms using domain knowledge in form of subsumption relations. Ontology-driven search is usually performed automatically by means of reasoning services handling particular aspects of an ontology representation language.

5.11 OntologyDomain

OntologyDomain	
Name Type Identifier	OntologyDomain `OntologyDomain`
Definition	Typically, the domain refers to established topic hierarchies such as the general purpose topic hierarchy DMOZ or the domain specific topic hierarchy ACM for the computer science domain
Comments	None

Table 5.95: Class: OntologyDomain

URI	
Name Type Identifier	URI `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	The URI of the ontology domain

Domain
Range
Cardinality
	`omv:OntologyDomain`
	`xsd:string`
	1:1

OMV version	2.1
Comments	None

Table 5.96: Property: URI

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	The name by which an ontology domain is formally known

Domain
Range
Cardinality
	`omv:Ontology`
	`xsd:string`
	1:1

OMV version	2.1
Comments	None

Table 5.97: Property: name

isSubDomainOf	
Name Type Identifier	isSubDomainOf `ObjectProperty`
Occurrence Constraint Category	`optional` Applicability information
Definition	Specifies the domain topic of which this domain topic is a sub domain
Domain Range Cardinality	`omv:OntologyDomain` `omv:OntologyDomain` 0:n
OMV version	0.8
Comments	Typically, the domain can refer to established topic hierarchies such as the general purpose topic hierarchy DMOZ or the domain specific topic hierarchy ACM for the computer science domain

Table 5.98: Property: isSubDomainOf

5.12 Party

Party	
Name Type Identifier	Party `class`
Definition	A party is a person or an organisation
OMV version	0.4
Comments	None

Table 5.99: Class: Party

isLocatedAt	
Name Type Identifier	isLocatedAt `ObjectProperty`
Occurrence Constraint Category	`optional` Availability Information
Definition	The geographical location of a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:Location`
	0:n

OMV version	0.9
Comments	None

Table 5.100: Property: isLocatedAt

developesOntologyEngineeringTool	
Name Type Identifier	developesOntologyEngineeringTool `ObjectProperty, inverseOf(toolDeveloper)`
Occurrence Constraint Category	`optional` Provenance information
Definition	References a tool developed by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:OntologyEngineeringTool`
	0:n

OMV version	0.6
Comments	None

Table 5.101: Property: developesOntologyEngineeringTool

developesOntologyLanguage	
Name Type Identifier	developesOntologyLanguage `ObjectProperty, inverseOf(languageDeveloper)`
Occurrence Constraint Category	`optional` Provenance information
Definition	References an ontology language developed by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:OntologyLanguage`
	0:n

OMV version	0.6
Comments	None

Table 5.102: Property: developeaOntologyLanguage

developesOntologySyntax	
Name Type Identifier	developesOntologySyntax `ObjectProperty, inverseOf(syntaxDeveloper)`
Occurrence Constraint Category	`optional` Provenance information
Definition	References an ontology syntax developed by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:OntologySyntax`
	0:n

OMV version	0.6
Comments	None

Table 5.103: Property: developesOntologySyntax

specifiesKnowledgeRepresentationParadigm	

Name
Type
 	
Identifier
	specifiesKnowledgeRepresentationParadigm
	`ObjectProperty`
	`inverseOf(knowledgeRepresentationParadigmSpecifiedBy)`

Occurrence Constraint Category	`optional` Provenance information
Definition	References an KR paradigm specified or defined by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:KnowledgeRepresentationParadigm`
	0:n

OMV version	0.6
Comments	None

Table 5.104: Property: specifiesKnowledgeRepresentationParadigm

definesOntologyType	
Name Type Identifier	definesOntologyType `ObjectProperty, inverseOf(definedBy)`
Occurrence Constraint Category	`optional` Provenance information
Definition	Reference an ontology type defined by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:OntologyType`
	0:n

OMV version	0.6
Comments	None

Table 5.105: Property: definesOntologyType

developesOntologyEngineeringMethodology	
Name Type Identifier	developesOntologyEngineeringMethodology `ObjectProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	References a ontology engineering method developed by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:OntologyEngineeringMethod`
	0:n

OMV version	0.6
Comments	None

Table 5.106: Property: developesOntologyEngineeringMethodology

specifiesLicense	
Name Type Identifier	specifiesLicense `ObjectProperty, inverseOf(licenseSpecifiedBy)`
Occurrence Constraint Category	`optional` Provenance information
Definition	References a license specified by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:LicenseModel`
	0:n

OMV version	0.6
Comments	None

Table 5.107: Property: specifiesLicense

hasAffiliatedParty	
Name Type Identifier	hasAffiliatedParty `ObjectProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	References another party that is affiliated with the instance
Domain Range Cardinality	`omv:Party` `omv:Party` 0:n
OMV version	0.2
Comments	None

Table 5.108: Property: hasAffiliatedParty

createsOntology	
Name Type Identifier	`createsOntology` `ObjectProperty, inverseOf(ontologyCreator)`
Occurrence Constraint Category	`optional` Provenance information
Definition	References an ontology created by a party

Domain
Range
Cardinality
	`omv:Party`
	`omv:Ontology`
	0:n

OMV version	0.7
Comments	None

Table 5.109: Property: createsOntology

contributesToOntology	

Name
Type
 	
Identifier
	`contributesToOntology`
	`ObjectProperty`
	`inverseOf(ontologyContributor)`

Occurrence Constraint Category	`optional` Provenance information
Definition	References an ontology a party made contributions to

Domain
Range
Cardinality
	`omv:Party`
	`omv:Ontology`
	0:n

OMV version	0.7
Comments	None

Table 5.110: Property: contributesToOntology

5.13 Person

Person	
Name Type Identifier	Person `class`
Definition	A named individual
OMV version	0.1
Comments	Represents an individual responsible for the creation, or contribution to an ontology

Table 5.111: Class: Person

lastName	
Name Type Identifier	lastName `DatatypeProperty`
Occurrence Constraint Category	`required` Provenance information
Definition	The surname of a person

Domain
Range
Cardinality
	`omv:Person`
	`xsd:string`
	1:1

OMV version	0.2
Comments	None

Table 5.112: lastName

firstName	
Name Type Identifier	firstName `DatatypeProperty`
Occurrence Constraint Category	`required` Provenance information
Definition	The first name of a person

Domain
Range
Cardinality
	`omv:Person`
	`xsd:string`
	1:n

OMV version	0.2
Comments	None

Table 5.113: firstname

eMail	
Name Type Identifier	email `DatatypeProperty`
Occurrence Constraint Category	`required` Provenance information
Definition	The email address of a person

Domain
Range
Cardinality
	`omv:Person`
	`xsd:string`
	1:n

OMV version	0.1
Comments	None

Table 5.114: eMail

phoneNumber	
Name Type Identifier	phoneNumber `DatatypeProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	The phone number of a person

Domain
Range
Cardinality
	`omv:Person`
	`xsd:string`
	0:n

OMV version	0.1
Comments	None

Table 5.115: Property: phoneNumber

faxNumber	
Name Type Identifier	faxNumber `DatatypeProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	The fax number of a person

Domain
Range
Cardinality
	`omv:Person`
	`xsd:string`
	0:n

OMV version	0.1
Comments	None

Table 5.116: faxNumber

isContactPerson	
Name Type Identifier	isContactPerson ObjectProperty, inverseOf(contactPerson)
Occurrence Constraint Category	optional Provenance information
Definition	Instance is contact person of an organisation
Domain Range Cardinality	omv:Person omv:Organisation 0:n
OMV version	0.7
Comments	None

Table 5.117: isContactPerson

5.14 Organisation

Organisation	
Name Type Identifier	Organisation `class, subclassOf(Party)`
Definition	An organisation of some kind
OMV version	0.6
Comments	Represents social institutions such as universities, companies, societies etc.

Table 5.118: Class: Organisation

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` Provenance information
Definition	The name by which an organisation is formally known

Domain
Range
Cardinality
	`omv:Organisation`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.119: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty` Used Identifier for this entity.
Occurrence Constraint Category	`required` Provenance information
Definition	A short name by which an organisation is known

Domain
Range
Cardinality
	`omv:Organisation`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 5.120: Property: acronym

hasContactPerson	
Name Type Identifier	`hasContactPerson` `ObjectProperty, inverseOf(ishasContactPerson)`
Occurrence Constraint Category	`optional` Provenance information
Definition	A contact person in the organisation

Domain
Range
Cardinality
	`omv:Organisation`
	`omv:Person`
	0:n

OMV version	0.6
Comments	None

Table 5.121: hasContactPerson

5.15 Location

Location	
Name Type Identifier	Location `class`
Definition	A location.
OMV version	0.9
Comments	The geographical location of a party. To keep things simple we use only DatatypeProperties instead of introducing classes for country, street, etc.

Table 5.122: Class: Location

state	
Name Type Identifier	state `DatatypeProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	The state of a country.

Domain
Range
Cardinality
	`omv:Location`
	`xsd:string`
	0:1

OMV version	0.9
Comments	None

Table 5.123: Property: state

country	
Name Type Identifier	country `DatatypeProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	The name of the country

Domain
Range
Cardinality
	`omv:Location`
	`xsd:string`
	0:1

OMV version	0.9
Comments	Changed the name from land to country

Table 5.124: Property: country

city	
Name Type Identifier	city `DatatypeProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	Name of the city (and zip code).

Domain
Range
Cardinality
	`omv:Location`
	`xsd:string`
	0:1

OMV version	0.9
Comments	None

Table 5.125: Property: city

street	
Name Type Identifier	street `DatatypeProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	Name of the street and number (address).

Domain
Range
Cardinality
	`omv:Location`
	`xsd:string`
	0:1

OMV version	0.9
Comments	None

Table 5.126: Property: street

Chapter 6

OMV Extensions

The OMV core metadata is intended to evolve towards a *commonly agreed* scheme for Semantic Web ontologies. In contrast to this ambitious goal we are aware that for specific domains, tasks or communities extensions in any direction might be required. These extensions should be compatible to the OMV core, but in the same time fulfill the requirements of a domain, task or community-driven setting.

The character of an OMV extension is a *metadata ontology* itself which imports the OMV core ontology. There are no restricting modelling guidelines to be met. However we provide a basic inventory of design decisions and guidelines, which are recommended to be applied for the extension modules (see Chapter 2).

Recalling the main metadata elements of the OMV core we envision the development of OMV extension ontologies elaborating the aspects these elements account for: detailed information about the knowledge representation field (represented in the OMV core by elements such as `KnowledgeRepresentationParadigm`, `OntologyLanguage` etc.), about the conceptual model of the ontology, about Ontological Engineering methodologies or about the various ways to evaluate ontologies. Further on, one might consider including additional extension modules, which are currently not covered by the OMV metadata scheme, but are related to the ontology field. For example OMV does not consider yet topics like Argumentation or Rating. Other topics not covered in OMV core like ontology management (Merging, Alignment, Versioning) were eliminated from the first version of the metadata schema as a result of the requirements analysis phase (see Chapter 3).

Some of the aforementioned extensions are currently being developed in collaboration or exclusively by partner institutions. In the following we provide a brief description of these initiatives.

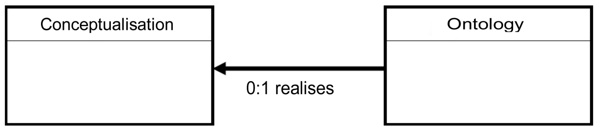

Figure 6.1: Relationship between conceptualisation and ontology

6.1 Conceptualisation extension

6.1.1 Conceptualisation vs. Ontology

This OMV extension focus on the conceptual model which is specified by an ontology, making a clear distinction between a **conceptualisation** and its **specification(s)** in concrete representation languages. This is based on a core principle of knowledge engineering, which clearly recommends the separation between knowledge and implementation level. Any existing ontology on the Web is the implementation of an (abstract) *conceptual model* i.e. the conceptualisation. For example, the intranet of a company might be conceptualized as a graph of workstations and servers and might be implemented using the OWL language. In this case the graph structure representing the intranet represents the *conceptualization*, which can be formalized in various way, from an abstract image in the mind of an individual to graphics of vertices and edges or natural languages. The conceptual model is implemented in a particular representation language and stored in a file system, giving birth to an *ontology*.

Therefore, the two concepts are defined as follows:

- **[Conceptualisation]:** A *Conceptualisation (C)* represents the (abstract) core model or idea behind an ontology. It describes the core properties of an ontology, independently from any implementation details. For a general illustration of the relationship between conceptualisations and ontologies, we refer to Figure 6.1.
- **[Ontology]:** An *Ontology (O)* represents a specific realization of a conceptualisation. Therefore, it describes properties of an ontology that are related to the realization or implementation.[1]

The distinction between the two concepts provides an efficient mechanism for the realization of several ontology management utilities, such as tracking several versions and the evolution flow of an ontology or handling different representations of the same knowledge model. Technically a conceptualisation and an ontology are modeled as two separate classes, with the relation `realises` from the ontology to the conceptualisation.

[1]Note that an ontology representation does not necessarily map to a physical document, as a Semantic Web ontology might be stored in multiple Web documents linked by `owl:import` statements.

This means that there may be many possible ontologies for one conceptualisation, but one ontology can `implement` only one conceptualisation.

From a theoretical point of view an instance of an ontology can not 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 the members of its engineering team, who are likely to have authored the conceptualization. Once an ontology is implemented the extraction of the underlying conceptual structure is realizable only by means of non-trivial re-engineering activities. However, automatically extracting syntactical properties of an existing ontology is quite simple. In this case a *minimal ontology* description would exist without a concrete conceptualisation.

Conceptualisation and ontology entries share many features. However, they have different meanings and are used for different purposes in an ontology engineering scenario. 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 ontology individual are quite similar (e.g. the same `party` as `creator`, the same `domain`). A realization of the same conceptualisation in OWL instead of RDF(S) would however generate a new ontology entry, which is comparatively different from the metadata information captured for the original conceptualisation.

6.1.2 Conceptualisation class

The class `Conceptualisation` captures information about the knowledge model of an ontology. It is shared by multiple `Ontologies` representing the model in particular representation languages.

Conceptualisation	
Name Type Identifier	Conceptualisation `class`
Definition	The knowledge model of an ontology
OMV version	0.9
Comments	Renamed from `OntologyBase` to `OntologyConceptualisation` and finally to `Conceptualisation`.

Table 6.1: Class: Conceptualisation

hasImplementation	
Name Type Identifier	hasImplementation `ObjectProperty, inverseOf{implements}`
Occurrence constraint Category	`optional` General information
Definition	Specifies the ontology(s) that realise this conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`omv:Ontology`
	0:n

OMV version	0.1
Comments	None

Table 6.2: Property: hasImplementation

name	
Name Type Identifier	name `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	The name by which a conceptualisation is formally known

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 6.3: Property: name

acronym	
Name Type Identifier	acronym `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	A short name by which the conceptualisation is formally known

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 6.4: Property: acronym

description	
Name Type Identifier	description `DatatypeProperty`
Occurrence Constraint Category	`required` General information
Definition	Free text description of the conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`xsd:string`
	1:1

OMV version	0.1
Comments	None

Table 6.5: Property: description

documentation	
Name Type Identifier	`documentation` `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	URL for further documentation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`xsd:string`
	0:1

OMV version	0.2
Comments	None

Table 6.6: Property: documentation

keywords	
Name Type Identifier	`keywords` `DatatypeProperty`
Occurrence Constraint Category	`optional` General information
Definition	List of keywords related to the conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`xsd:string`
	0:n

OMV version	0.1
Comments	

Table 6.7: Property: keywords

hasLicense	
Name Type Identifier	hasLicense `ObjectProperty`
Occurrence Constraint Category	`optional` Availability information
Definition	Information about the underlying license model

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`omv:LicenseModel`
	0:1

OMV version	0.1
Comments	Reference to a concrete LicenseModel.

Table 6.8: Property: hasLicense

hasDomain	
Name Type Identifier	hasDomain `ObjectProperty`
Occurrence Constraint Category	`optional` Applicability information
Definition	Specifies the domain topic of the conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`omv:OntologyDomain`
	0:n

OMV version	0.8
Comments	Typically, the domain can refer to established topic hierarchies such as the general purpose topic hierarchy DMOZ or the domain specific topic hierarchy ACM for the computer science domain

Table 6.9: Property: hasDomain

isOfType	
Name Type Identifier	isOfType `ObjectProperty`
Occurrence Constraint Category	`required` Applicability information
Definition	The nature of the content of the conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`omv:OntologyType`
	1:1

OMV version	0.1
Comments	See section 5.2 for details. The conceptualisation has a type?

Table 6.10: Property: isOfType

hasContributor	
Name Type Identifier	hasContributor `ObjectProperty`
Occurrence Constraint Category	`optional` Provenance information
Definition	Contributor to the conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`omv:Party`
	0:n

OMV version	0.1
Comments	None

Table 6.11: Property: hasContributor

hasCreator	
Name Type Identifier	hasCreator `ObjectProperty`
Occurrence Constraint Category	`required` Provenance information
Definition	The main party responsible for the conceptualisation

Domain
Range
Cardinality
	`omv:Conceptualisation`
	`omv:Party`
	1:n

OMV version	0.1
Comments	None

Table 6.12: Property: hasCreator

isConceptualExtension	
entity Name Type Identifier	isConceptualExtension `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	A conceptualisation which is a conceptual extension of another conceptualisation
Domain Range Cardinality	`omv:Conceptualisation` `omv:Conceptualisation` 0:1
OMV version	0.9.1
Comments	Sub-property of `conceptualRelation`

Table 6.13: Property: isConceptualExtension

hasConceptualIntegration	
entity Name Type Identifier	hasConceptualIntegration `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	A conceptualisation which conceptually integrates other conceptualisation(s)
Domain Range Cardinality	`omv:Conceptualisation` `omv:Conceptualisation` 0:n
OMV version	0.9.1
Comments	Sub-property of `conceptualRelation`

Table 6.14: Property: hasConceptualIntegration

hasConceptualRelation	
entity Name Type Identifier	hasConceptualRelation `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship Information
Definition	A conceptualisation which is conceptually similar to other conceptualisation(s)
Domain Range Cardinality	`omv:Conceptualisation` `omv:Conceptualisation` 0:n
OMV version	0.9.1
Comments	Super-property of the remaining conceptual relationships

Table 6.15: Property: hasConceptualRelation

isConceptualSpecialisation	
entity Name Type Identifier	isConceptualSpecialisation `ObjectProperty`
Occurrence Constraint Category	`optional` Relationship information
Definition	A conceptualisation which conceptually specialises another conceptualisation
Domain Range Cardinality	`omv:Conceptualisation` `omv:Conceptualisation` 0:1
OMV version	0.9.1
Comments	Sub-property of `conceptualRelation`

Table 6.16: Property: isConceptualSpecialisation

isConceptualGeneralisation	
entity Name	isConceptualGeneralisation
Type	`ObjectProperty`
Identifier	Used Identifier for this entity.
Occurrence Constraint	`optional`
Category	Relationship information
Definition	A conceptualisation which conceptually generalises another conceptualisation
Domain	`omv:Conceptualisation`
Range	`omv:Conceptualisation`
Cardinality	0:1
OMV version	0.9.1
Comments	Sub-property of `conceptualRelation`

Table 6.17: Property: isConceptualGeneralisation

Chapter 7

Using Metadata

Metadata information can be embedded in a variety of ways in the majority of ontology management services proposed so far. In this document we focus on the main usage scenarios.

We identified the following roles w.r.t. developing and deploying ontology metadata and ontologies:

- **Ontology developer** - The party primarily responsible for developing an ontology. They are expected to provide the majority of the metadata information and might use ontology metadata during ontology reuse processes.
- **Ontology contributor** - A party involved in an ontology development process. They are expected to create and use metadata in the same manner as the previous category.
- **Ontology reviewer** - A party responsible for evaluating an ontology. Ontology metadata information provides them with a useful means to ease the evaluation process. Further on, reviewers are expected to report on the reviewing process and its results in form of an entry in the metadata extension module for evaluation.
- **Ontology user** - A party applying an ontology for a specific purpose. They are expected to provide information about the application scenario in the appropriate extension module.

Chapter 8

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[1]) and centralized (ONTHOLOGY[2]) sharing of ontology metadata based on OMV .

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.

Acknowledgements

This proposal is based on a huge number of discussions and many helpful arguments by persons from academia and industry. Especially we would like to thank our colleagues York Sure (AIFB), M. Carmen Suárez-Figueroa (UPM), Peter Haase (AIFB), Asunción Gómez-Pérez (UPM), Denny Vrandecic(AIFB) and Rudi Studer (AIFB). Furthermore, we thank our partners from the EU project Knowledge Web for their supporting discussions.

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project Knowledge Web, number IST-2004-507482. The partners in this project are: University of Innsbruck (UIBK, Coordinator), Ècole Polythechnique Fédérale de Lausanne (EPFL), France Telecom (FT), Freie Universität Berlin (FU Berlin), Free University of Bozen-Bolzano (FUB), Institut National de Recherche en Informatique et en Automatique (INRIA), Centre for Research

[1]http://ontoware.org/projects/oyster
[2]http://www.onthology.org

and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH), Learning Lab Lower Saxony (L3S), National University of Ireland Galway (NUIG), The Open University (OU), Universidad Politécnica de Madrid (UPM), University of Karlsruhe (UKARL), University of Liverpool (UniLiv), University of Manchester (UoM), University of Sheffield (USFD), University of Trento (UniTn), Vrije Universiteit Amsterdam (VUA), Vrije Universiteit Brussel (VUB).

Bibliography

[1] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (uri): Generic syntax, 1998.

[2] A. Gangemi, D. M. Pisanelli, and G. Steve. An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies. *Data Knowledge Engineering*, 31(2):183–220, 1999.

[3] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2003.

[4] N. Guarino. Formal Ontology and Information Systems. In *Proceedings of the FOIS'98*, pages 3–15, 1998.

[5] M. Klein and D. Fensel. Ontology versioning for the semantic web, 2001.

[6] O. Lassila and D. McGuinness. The role of frame-based representation on the semantic web. KSL Tech Report Number KSL-01-02, 2001.

[7] A. Lozano-Tello and A. Gomez-Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15(2), 2004.

[8] D. L. McGuinness. Ontologies Come of Age. In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.

[9] National Information Stadards Organization. Understanding metadata. NISO Press, 2004.

[10] E. Paslaru Bontas, M. Mochol, and R. Tolksdorf. Case Studies on Ontology Reuse. In *Proceedings of the IKNOW05 International Conference on Knowledge Management*, 2005.

[11] H. S. Pinto and J. P. Martins. A methodology for ontology integration. In *Proc. of the International Conf. on Knowledge Capture K-CAP01*, 2001.

[12] T. Russ, A. Valente, R. MacGregor, and W. Swartout. Practical Experiences in Trading Off Ontology Usability and Reusability. In *Proc. of the Knowledge Acquisition Workshop (KAW99)*, 1999.

[13] 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/`.

[14] M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology Reuse and Application. In *Proc. of the Int. Conf. on Formal Ontology and Information Systems FOIS98*, 1998.

[15] Y. Wand and R. Weber. Information Systems and Conceptual Modelling: A Research Agenda. *Information Systems Research*, 13(4), 2002.