OntoMap: Ontologies for Lexical Semantics

Atanas K. Kiryakov and **Kiril Iv. Simov** and **Marin Dimitrov**
OntoText Lab., Sirma AI Ltd.
Hr. Botev 38A, Sofia 1000, Bulgaria, {naso,marin}@sirma.bg
Linguistic Modelling Laboratory, Bulgarian Academy of Sciences
Acad. G. Bonchev Str. 25A, 1113 Sofia, Bulgaria, kivs@bgcict.acad.bg

Abstract

The upper-level ontologies are theories that capture the most common concepts, relevant to many tasks. These ontologies are used to represent the skeleton of the human common-sense in such a formal way that covers as many aspects of the knowledge as possible. The result often is a complex and abstract philosophic theory.

Currently the evaluation of the feasibility of such ontologies is very expensive mostly because of technical problems such as different representations and terminologies used. Additionally, there are no formal mappings between the upper-level ontologies that could make their understanding, study, and comparison. As a result, the upper-level models are not widely used easier. We present OntoMap (http://www.ontomap.org) — a project with the pragmatic goal to facilitate an easy access, understanding, and reuse of such resources. It is also a good framework for research on lexical phenomena such as systematic polysemy.

1 Introduction

Understanding a text can require a vast amount of knowledge about the world including common-sense knowledge, factual knowledge about the context of the writing of the text and general knowledge. We can think about the process of understanding as a construction of a knowledge base representing the content of the text. Of course, such a view is an oversimplification of what real understanding is, but in our opinion this is a good starting point. The knowledge base connected to a text can be divided into two parts corresponding to the common understanding that a knowledge base comprises an ontology part (*ontology content*) and a specific knowledge base part (*factual content*). In general, each text represents both kinds of knowledge. The ontology part of the text constrains the interpretations of the content of the text within the context of some domain(s) and the factual knowledge part asserts what state of affairs is the text about. We think that a lexical knowledge base (like WordNet) can be used as a key tool for constructing the ontology content of a text, especially if it is linked to a world knowledge base (like Cyc). In this paper we present the current state of OntoMap project which has the pragmatic goal to facilitate an easy access, understanding, and reuse of such resources. Using the OntoMapO we represent a place where various upper-level models can be used together in a single framework which is a very common task when building lexical resources or studying lexical phenomena such as systematic polysemy. In this paper we present the tool for comparison and mapping of ontologies.

The structure of the paper is as follows: the next section is an introduction to the Upper-Level Ontologies. Section 3 discusses the representation languages and primitives. Then we describes the primitives used in the OntoMap project ordered in the OntoMapO ontology. The following section focuses on the methodology for mapping concepts between ontologies. The initial set of ontologies to be hosted and the mappings between them are discussed in section 6. An example of the usability of OntoMap is given in section 7. The last section concludes the paper.

2 Upper-Level Ontologies

The upper-level ontologies capture mostly concepts that are basic for the human understanding of the world. They are "grounded" in (supported by, wired to) the common sense that makes it hard to formalize a strict definition for them. They represent the so called prototypical knowledge. For example, what should be a formal KL-ONE-style or Frame-style definition of a "table". Most of the tables have 4 legs, however, there are pretty obvious exceptions for tables with three legs, a single leg or even without anything to be considered as a leg. There could be also a "serious" table with 6 legs. What should be the minimum and maximum cardinality for the slot/role leg? And what should be the type restriction? This is the reason to have most of the upper-level concepts being primitive in KL-ONE terms – they can only have partial definitions, some necessary conditions that involve other partially defined concepts. This is the practical reason to have the upper-level ontologies (for example, Upper Cyc Ontology, Sensus, MikroKosmos) defined mainly in terms of taxonomic relations.

2.1 Domain-Specific vs. Upper-Level Ontologies

This pseudo-dilemma seems to be mostly a question of goal and scope rather than a representational or management problem. However, there exists a significant real difference between the two types of ontologies. The domain-specific ontologies that are trying to capture, for example, a market segment or certain scientific area typically consist of well-defined concepts. For example, in the natural sciences (Mathematics, Physics, Chemistry, Biology, Medicine) the knowledge is usually easy to formalize because it is more or less systematic — it could be expressed using well-defined scientific terms. In such cases, the objects in the universe of discourse are either purely abstract or they are some idealized/simplified models of the real phenomena in the world. For instance, A triangle is nothing more than a polygon with three angles.

2.2 Lexical Knowledge Bases

The so-called lexical knowledge bases (LKB, such as WordNet) are lexicons, thesaurus, or dictionaries that attempt to formalize the lexical semantics — the meanings of the words in one or more natural languages. Similar to the upper-level concepts, the meanings of the words are grounded in the common understanding of huge populations — there are no formal definitions, the words can bear a number of different meanings often based on associations, typical uses, collocations, and prototypical knowledge. Going further, the meanings of many words are just primitive concepts. Historically the LKBs and the upper-level ontologies seriously influenced each other. Some upper-level ontologies were developed on the basis of a LKB — such example is the SENSUS ontology ((Knight & Luk 1994)). Other upper-level ontologies were developed in order to give formal semantics to a LKB — such an example is the EuroWordnet Top Ontology, (Vossen 1999). This is the reason to have a number of LKB semantic resources included in the initial set of ontologies to be hosted in OntoMap.

2.3 Philosophical diversity

The existence of several upper-level ontologies that disagree on the most basic concepts about the entities in the world demonstrates a significant philosophical diversity. The practical goals OntoMap project is after seem to require clarification of these basic discrepancies. Which properties of the entities in the world are the most basic ones? What follows from different choices on this level? On which level of generality the differences disappear if they disappear at all? Our understanding is that OntoMap should not try to choose the best upper-model or to produce a new one, but just to facilitate the comparison of such ontologies.

2.4 Some Disadvantages of the Automatic Mapping

Automatic mapping or merging of ontologies is not involved in OntoMap — we start with the assumption that there exist some handcrafted mappings, or such can be developed. Even though it seems that automatic mapping could reduce the efforts, the typical heuristics employed (see (McGuinness et. al. 2000), (Campbell & Shapiro 1998)) can have a very limited role in the case of upper-level ontologies because of a number of reasons:

- there is relatively small number of upper-level resources, because they are complex, expensive, and (potentially) more reusable than the domain-specific ones;

- they are more complex than the domain-specific ones, because they handle more abstract and partially defined concepts. Much of the semantics is represented as a free text, instead of a formula. So, in many cases the equivalence between two concepts could be judged only by interpretation of text glosses;

- the mappings between upper-level ontologies are more re-usable because the ontologies are more reusable;

- the quality of the mapping is extremely important because a mistake in the upper-levels can have terrible effect on the lower levels.

3 Unified Representation Needed

In order to provide a uniform representation of the ontologies and the mappings between them, a relatively simple meta-ontology (let us call it OntoMapO) of property types and relation-types should be defined. Before presenting OntoMapO we will make an overview of the related problems and approaches.

3.1 Terminological Diversity

There are number of different notions (or terminologies) that are currently used in knowledge management community. The differences (both phraseological and conceptual) are rooted in the main paradigms in the knowledge representation. Here is a non-exhaustive overview of the most popular "languages" used by the ontologists (for an extended comparison see (Kiryakov et al. 2001)):

- **Concepts, Relations, Properties** — terms usually inspired by the **semantic networks**, mathematics and philosophy;
- **Classes, Slots, Facets and Frames** — the frame-based terminology;
- **Concepts, Roles, Individuals** — the terminology used in the so called description logics (DL);
- **Classes, Objects, Attributes** — the terminology used in the **object-oriented** paradigm (OO), developed mainly for the purposes of the software engineering;
- **Collections, Individuals, Predicates, Constants** — the terminology used in the Cyc knowledge base.

3.2 The Conceptual Level

There are number of attempts to resolve the terminological diversity by managing ontologies in a representation-independent fashion on the so called knowledge-level or conceptual level. Two of the most popular approaches are reported in (Gomez-Perez et al. 1998) (ODE) and (Maedche et. al. 2000) (OntoEdit). Even sticking to the frame-based terminology the knowledge-model of Protégé-2000 (see (Noy et. al. 2000)) is also a good example for a self-contained and well designed conceptualization that provides sufficient expressive power to capture ontologies encoded in different languages.

A comprehensive classification of the different kinds of properties is reported in (Guarino & Welty 2000) — according to different combinations of the meta-properties *indentity*, *rigidity* and *dependence* it introduces seven different notions corresponding to "Concept" in ODE. The primitives used in Cyc (see (Cyc 1997)) are interesting at least because the approach is proven in a really large-scale knowledge base.

4 OntoMapO: The OntoMap primitives

OntoMap is trying to use the minimal useful set of primitives. We are led by the understanding that the oversimplification is not that fatal for the overall usability as a complex system of primitives could be. We tried to design OntoMapO to capture most of the semantics usually encoded in upper-level models. The guideline was to give access to 80% of the content of of the upper-level ontologies within 20% of the complexity of the representation needed to capture everything.

We developed a minimalistic meta-ontology that is as self-describing as possible. Thus, most of the primitives are defined just in terms of the rest of the OntoMapO primitives.

OntoMapO ontology could be understood also as a language. A simple language that provides some expressive power via single kind of expressions – binary relations between concepts. We are intentionally not providing specific syntax in order to keep it as representation independent as possible. Further in this paper we will use a LISP-like syntax to serialize the relations. However, it is obvious that many other notations (say XML) could perform equally well.

Thus we undertake an approach opposite to the one employed in Ontolingua, (Gruber 1991), following the rationale that even though many distinctions could be clearly defined in Ontolingua the most of the semantic-model developers cannot understand them. Our vision is that the database designers, for example, should not be expected to learn complex frame-based theories. Each concept is represented in Ontolingua with about twenty slots, some of which are not obvious for people that do not understand frames. For example, if somebody wants to understand the definition of `Corporation` in the Enterprise Ontology as it is represented in Ontolingua (see (Uschold 1996) and (Uschold 1998)) s/he has to bother about the meaning of slots like `Set-Cardinality` and `Relation-Universe`.

OntoMapO includes the following primitives:

4.1 Concepts, Relations, and Ontologies

Concept is the most basic primitive, so, we are leaving it to the reader's intuition. Just as a reference point the concepts could be compared to the constants in Cyc. The concepts could be related to each other by *binary relations*. Each binary relation has a type that is a concept. Each concept belongs to an ontology and, of course, there could be many different ontologies.

4.2 Instantiation in addition to inheritance

Our semantic framework got some inspiration form Cyc, Protégé-2000, and RDFS representation models (see (Cyc 1997), (Noy et. al. 2000), and (RDF 1999)) — in addition to the inheritance relations we also employ as a basic mechanism the instantiation. So, the concepts are not only described by their parents and children in the subsumption hierarchy but also form the classes that they belong to. The classes themselves are also concepts that could belong to other classes and so on. In this way an infinite number of meta-levels could be defined.

We will use a simple set-theoretical semantics to explain the distinction between the inheritance and instantiation. Suppose that each concept is interpreted as a set of its instances. So, `(InstanceOf I C)` means that $\mathtt{I} \in \mathtt{C}$. In the same fashion `(ChildOf C1 C2)` means that $\mathtt{C1} \subset \mathtt{C2}$. This interpretation has some pretty reasonable consequences:

- the inheritance relations are transitive — if `(ChildOf C1 C2)` and `(ChildOf C2 C3)` it follows that `(ChildOf C1 C3)`. Really, from $\mathtt{C1} \subset \mathtt{C2}$ and $\mathtt{C2} \subset \mathtt{C3}$ it follows that $\mathtt{C1} \subset \mathtt{C3}$

- the instantiation is non-transitive — if $\mathtt{I} \in \mathtt{C}$ and $\mathtt{C} \in \mathtt{MetaC}$ it does **not** follow that $\mathtt{I} \in \mathtt{MetaC}$

- the instantiation is transitive with respect to inheritance — if `(InstanceOf I C1)` and `(ChildOf C1 C2)` it follows that `(InstanceOf I C2)`. Really, from $\mathtt{I} \in \mathtt{C1}$ and $\mathtt{C1} \subset \mathtt{C2}$ it follows that $\mathtt{I} \in \mathtt{C2}$

An obvious advantage of such extensive use of instantiation is that it makes the hierarchy less tangled avoiding multiple-inheritance on many places. As a design principle, instantiation should be used to express non-sortal properties of the concepts (see (Guarino & Welty 2000)). For example, in OntoMapO (see below) we represent the transitivity of a relation type (say, `ChildOf`) via instantiation. It happens due to the following reason: the fact that certain relation is transitive does not determine its identity — it is just a rigid property, namely a Category for relations.

4.3 Relations

Each relation between two concepts is an instance of the concept representing its type. Let us extend the set-theoretical interpretation of our model — if `(RelA B C)` then the pair $<\mathtt{B},\mathtt{C}> \in \mathtt{RelA}$. Suppose there are two concepts `RelA` and `RelB` that represent relation types and the first one inherits the second one `(ChildOf RelA RelB)`. Our interpretation correctly predicts that `RelB` holds between all concepts where `RelA` holds. Let us show how it works:

- let have concepts `A` and `B` and there is a relation of type `RelA` between them `(RelA A B)`

- following our interpretation we can state that $<\mathtt{A},\mathtt{B}> \in \mathtt{RelA}$

- also `(ChildOf RelA RelB)` means that $\mathtt{RelA} \subset \mathtt{RelB}$

- now it is obvious that $<\mathtt{A},\mathtt{B}> \in \mathtt{RelB}$, that means that

- there is a relation of type `RelB` between the concepts `A` and `B`.

In OntoMapO all the concepts representing relation types should be instances of the `BinaryRel` concept or at least one of its children. Further, OntoMap inference engine considers a binary relation to be transitive iff it is an instance of `TransitiveRel` that is a child of `BinaryRel`. Examples for transitive relation are `ChildOf` and `Equivalent`. Analogously, a concept represents a symmetric relation type iff it is an instance of `SymmetricRel` — we can take `Inverse` relation (discussed below) as such example.

4.4 Inverse Relations

Another principle that we followed was to define an inverse relation for each of the OntoMapO relations except the symmetric ones, of course. The rationale behind this was two-fold:

- to emphasize that the OntoMap relations (in contrast to the slot notion, for example) does not give any representational preference to the concept in the first place
- make the relations easy to read and follow in both directions

So, `ChildOf` relation has its inverse `ParentOf` relation; `InstanceOf` is inverse to `ClassOf`. In order to keep some correspondence to the frame-based systems we defined `HasSlot` relation as an inverse to the `Domain` relation that could be defined between a relation type and the concept which instances could be first arguments of the relation. Analogously, `Reifies` is inverse to the `Range` relation that holds between a relation type and a concept which instances could be second arguments of the relation. Here are some real constraints that take place in OntoMapO:

- `(Domain Inverse BinaryRel)` and the equivalent statement that `(HasSlot BinaryRel Inverse)`
- `(Range Inverse BinaryRel)`
- `(Domain ChildOf Concept)` and its equivalent `(HasSlot Concept ChildOf)`

4.5 Predefined Relations

Let us call *sub-relations* of a relation `R` all its direct or indirect children as well as the relations that are equivalent to it or one of its sub-relations.

OntoMap considers as an equivalence relation each relation that is sub-relation `Equivalent`. In a similar fashion, all the sub-relations of `ChildOf` and `ParentOf` are treated as inheritance relations. Analogously, one relation is an instantiation relation iff it is a sub-relation of `InstanceOf` or `ParentOf` relations. Obviously, all the sub-relations of `Inverse` are properly interpreted as inversion by the OntoMap inference engine.

This approach makes the basic primitives that the OntoMap inference engine understands extensible. For example, when explaining Cyc's knowledge model to OntoMap it is easy to define that `(Equivalent #$genls ChildOf)` — this way OntoMap automatically starts to understand this kind of Cyc inference relations without any need to translate them further.

4.6 The Hierarchy

The hierarchy below is basically an inheritance tree augmented with some instantiation information — after each concept name in brackets we have the (most specific) classes it belongs to.

```
Top (Concept)
  Concept (Concept)
    BinaryRel (Concept)
      TransitiveRel (Concept)
      SymmetricRel (Concept)
  Ontology (Concept)
    Module (Concept)
  ChildOf (TransitiveRel)
    MuchMoreSpecific (TransitiveRel)
  ParentOf (TransitiveRel)
    MuchMoreGeneral (TransitiveRel)
  ClassOf (BinaryRel)
    ExactClassOf (BynaryRel)
  InstanceOf (BinaryRel)
    TopInstanceOf (BinaryRel)
  SimilarTo (TransitiveRel,
                      SymmetricRel)
    Equivalent(TransitiveRel,
                      SymmetricRel)
  Inverse (SymmetricRel)
  ChildAsClass (BinaryRel)
  ParentAsInstance (BynaryRel)
  Domain (BinaryRel)
  Range (BinaryRel)
  HasSlot (BinaryRel)
  Reifies (BinaryRel)
  DisjointWith (BinaryRel)
```

5 Ontology-Mapping Primitives

There is no formal difference between the relations that can be used inside an ontology and those to be used for mapping of concepts in different ontologies. However there are number of relations that are not expected to be used inside well defined ontology — those should be used for handling structural differences between different ontologies. For example, the `(TopInstance A B)` should be used when a concept `A` from one ontology exists as a concept `B` on the upper level of denotation in another ontology, i.e. `(ChildOf X`

A) holds iff (InstanceOf X B) holds. Such design patterns should not be tolerated inside a single ontology. Here follow explanations for these relations:

- `MuchMoreSpecific`, `MuchMoreGeneral` – the first concept is much more specific (resp. general) than the second one. Both are transitive and inverse to each other and have to be used to "constrain" the meaning of a concept that doesn't have an equivalent or even similar concept in another ontology;
- `TopInstance` – the first concept is the most general instance of the second one. Inverse to `ExactClass`;
- `ExactClass` – the first concept is a kind of meta-concept, the second concept is the most general instance of the first one. Inverse to `TopInstance`;
- `ParentAsInstance` – the first concept is more general than all the instances of the second one that is a meta-concept. Inverse to `ChildAsClass`
- `ChildAsClass` – the first concept is a meta-concept (class), all its instances are more specific than the second concept. Inverse to `ParentAsInstance`

6 The Initial Set of Ontologies

The following ontologies will be hosted initially: Upper Cyc Ontology [UCYC]; EuroWordnet Top Ontology; EuroWordnet Clusters – the clusters of EWN base concepts classified by top concepts, an extension of EWNTOP; WordNet 1.5, 1.6 and 1.7 [WNUB7] unique beginners and top nouns; CORELEX; SIMPLE Core Ontology; MikroKosmos top-level [MKOSTOP]; SENSUS top-level [SENSTOP].

For each of the ontologies there will be available an "executive summary" as well as the most important documents about it (papers, reports, guides and so on), URLs. Of course, the original "distributives" provided by the creators will be also available. The following ontologies are already hosted on OntoMap: EWNTOP, WNUB7, SENSTOP, and MKOSTOP.

Mappings between some of the ontologies will be provided in order to ensure an easier understanding and comparison between them. So, the ontologies hosted will form an inter-connected graph. Such mapping already exists between EWNTOP and UCYC (see (Kiryakov & Simov 2000)). The mapping between WNUB7 and MKOSTOP and the later two ontologies are currently in development.

7 An Usability Example

Here follows an example of how the OntoMap could help understanding a complex case in the Upper Cyc Ontology - the `#$MeetingTakingPlace` constant. The comprehension comes from the following sources: it is deeply positioned in the tangled subsumption hierarchy; and also some important information is encoded via instantiation. The most readable representation in (Cyc 1997) is:

> The collection of human meeting events, in which #$Persons gather intentionally at a location in order to communicate or share some experience; business is often transacted at such a meeting. Examples include: a particular conference, a business lunch, etc.
> **isa**: #$DefaultDisjointScriptType, #$ScriptType, #$TemporalObjectType
> **genls**: #$SocialGathering
> some subsets: (16 unpublished subsets)

The underlined text represents hyper-references to descriptions of the appropriate constants. Below follows the standard view on the same concept provided by OntoMap:

Concept: #$MeetingTakingPlace [UpperCyc]
Gloss: The collection of human meeting events, ...
Super-concepts (parents): #$SocialGathering;
Indirect: #$IntangibleIndividual, #$CompositePhysicalAndMentalEvent, #$TemporalThing, #$PhysicalEvent, #$MentalEvent, #$Intangible, #$Thing, #$SpatialThing, #$MentalActivity, #$PurposefulAction, #$Situation, #$HumanActivity, #$AnimalActivity, #$Event, #$Action, #$Individual, #$SocialOccurrence
Indirect parents in other ontologies: Physical[EWN_Top], Top[EWN_Top], Mental[EWN_Top], Social[EWN_Top], 2ndOrderEntity[EWN_Top]
Instance of: #$DefaultDisjointScriptType #$TemporalObjectType
Indirect: #$Collection, #$ObjectType, #$Thing, #$SituationType, #$SetOrCollection, #$Intangible, #$MathematicalOrComputationalThing, #$ScriptType
Sub-concepts (children): none
Direct instances: none

All direct relations:
#$genls: #$SocialGathering
#$isa: #$TemporalObjectType
#$isa: #$DefaultDisjointScriptType

This was a "snap-shot" of the current on-line interface of OntoMap. Pay attentionto the fact that both indirect parents and classes are displayed and it is extremely useful — it requires serious efforts to reconstruct these indirect relations manually. Also, super-concepts in the EWN Top Ontology can be seen, and that provides a good impression about a possible position of `#$MeetingTakingPlace` there. So people that are familiar with EWN top can get an idea about the meaning of the Cyc constant. The `Meeting` word is an interesting example of polysemy.

8 Conclusion

OntoMap project is still in an early phase and it is hard to evaluate it. The experience gathered providing the Upper Cyc Ontology as MS Access database is encouraging – even though the original resource is available for a long time more than two hundred people found it useful and downloaded it in this shape just for few months. We got a very positive feedback for another experiment of ours – developing a mapping between the Upper Cyc Ontology and the EuroWordnet Top Ontology and then providing it as a database as well as an online service. Even without significant theoretical innovation such facilitatory efforts seem to be important for the research community.

Behind the web-site we implemented a Java-written inference engine that supports the OntoMapO language – it is sound and complete with its support for inheritance, instantiation, inverse, transitive, and symmetric relations. The biggest ontology that we experimented with (Upper Cyc Ontology, about 3000 concepts) can be loaded in a second and then queried in real-time. To get further information, visit `http:\\www.ontomap.org`

References

(Campbell & Shapiro 1998) Campbell, A.E. and Shapiro, S.C., *Algorithms for Ontological Mediation*, Technical Report 98-03, Dep. of CS and Engineering, State Univ. of New York at Buffalo, 1998.

(Cyc 1997) Cyc Ontology Guide: Introduction. `http://www.cyc.com/cyc-2-1/intro-public.html`

(Gomez-Perez et al. 1998) Gomez-Perez, A.; Fernandez, M.; Blazquez, M.; Garcia-Pinar, J. M. *Building Ontologies at the Knowledge Level using the Ontology Design Environment.* http://delicias.dia.fi.upm.es/articulos/ode/ode.html

(Gruber 1991) Gruber, Thomas R. *Ontolingua: A Mechanism to Support Portable Ontologies.* Technical Report KSL 92-66, Knowledge System Laboratory, Stanford University, 1991.

(Guarino & Welty 2000) Guarino, Nicola and Welty, Christopher *A Formal Ontology of Properties.* In France. R. Dieg and O. Corby (Eds.): EKAW 2000, LNAI 1937, pp. 97-112, Springer Verlag, 2000.

(Kiryakov et al. 2001) Kiryakov, Atanas; Simov, Kiril Iv.; Dimitrov, Marin. *TR1. OntoMap: The Upper-Ontology Portal.* OntoText Lab. technical report.

(Kiryakov & Simov 2000) Kiryakov, Atanas; Simov, Kiril Iv. *Mapping of EuroWordnet Top Ontology to Upper Cyc Ontology.* In: Proc. of "Ontologies and Text" workshop, EKAW 2000. Juan-les-Pins, French Riviera, Oct. 2, 2000. `http://www.ontotext.com/publications/`

(Knight & Luk 1994) Knight, K. and Luk, S. *Building a Large Knowledge Base for Machine Translation.* Proc. of the American Association of Artificial Intelligence Conference. Seattle, WA, 1994.

(McGuinness et al. 2000) McGuinness, Deborah L., Richard Fikes, James Rice, and Steve Wilder, *An Environment for Merging and Testing Large Ontologies*, Proc. of the 7th Intern. Conf. on Principles of Knowledge Representation and Reasoning. Breckenridge, Colorado, USA. April 12-15, 2000.

(Maedche et al. 2000) Maedche, A.; Schnurr, H.-P.; Staab, S.; and Studer, R. *Representation Language-Neutral Modeling of Ontologies.* In: Frank (ed.), Proc. of the German Workshop "Modellierung" 2000. Koblenz, Germany, April, 5-7, 2000.

(Noy et al. 2000) Noy, Natalya F.; Fergerson, Ray W.; Musen, Mark A. *The Knowledge Model of Protege-2000: Combining Interoperability and Flexibility.* In R. Dieg and O. Corby (Eds.): EKAW 2000, LNAI 1937, pp. 97-112, Springer Verlag, 2000.

(Vossen 1999) Vossen, Piek (ed.) *EuroWordNet General Document Version 3, Final*, July 19, 1999.

(RDF 1999) World Wide Web Consortium; Brickley, Dan; Guha, R.V. (eds.) *Resource Description Framework (RDF) Schema Specification*, 1999. `http://www.w3.org/TR/1998/WD-rdf-schema/`

(Uschold et al. 1998) Uschold, Mike; King, Martin; Moralee, Stuart; and Zorgios, Yannis *The Enterprise Ontology*, The Knowledge Engineering Review, 1998, Vol. 13.

(Uschold 1996) Uschold, Mike *Converting an Informal Ontology into Ontolingua: Some Experiences*, Univ. Edinburgh, Artificial Intelligence Application Institute, AIAI-TR-192, March 1996.