OntoMap: Upper-Ontology Service Agent

Marin Dimitrov
OntoText Lab., Sirma AI Ltd.
Chr. Botev 38A
Sofia 1000, Bulgaria
marin@sirma.bg

Atanas Kiryakov
OntoText Lab., Sirma AI Ltd.
Chr. Botev 38A
Sofia 1000, Bulgaria
naso@sirma.bg

Kiril Iv. Simov
Linguistic Modelling Lab.
Bulgarian Academy of Sci.
Acad. G. Bontchev Str. 25A
Sofia 1113, Bulgaria
kivs@bgcict.acad.bg

ABSTRACT

Domain-specific ontologies or models are needed for different tasks. These are typically developed from scratch that makes them hardly extensible and inconsistent with each other. Having the domain-specific models grounded in upper-level ontologies makes them more consistent, reusable (see [6]) and easier for integration within distributed and heterogeneous systems.

Currently the evaluation of upper-level resources is an expensive task 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 enhance any kind of studies and comparison. As a result, the upper-level models are not actually used, except some attempts in the NLP and IR applications. The goal of the OntoMap project is to facilitate an easy access, understanding, and reuse of general-purpose ontologies and upper-level models.

A semantic framework (OntoMapO) is presented that is small and easy enough to be learned on-the-flight. We tried to design it to be capable to capture most of the semantic usually encoded in upper-level models.

Technically the result will be `http://www.OntoMap.org` web-site that provides a number of upper-level ontologies and mappings between them in a number of different formats, including an application server giving an uniform access to the resources via widely accepted standards such as a FIPA compliant ontology service agent, OKBC (on a semantic and pragmatic level) and RMI, SOAP, and CORBA (on protocol level). This way OntoMap will be part of the semantic web, i.e. machine-understandable (see [4]) rather just human-readable. The semantic web makes the WWW a better place for artificial agents — OntoMap could be seen as a semantic dictionary or a translator that will facilitate these agents to unify their vision to the real world and collaborate. Isn't it pitty when two agents, for example, misunderstanding each other because the first one uses rectangular coordinates while the second one uses polar?

1. INTRODUCTION

The next section presents an introduction to the Upper-Level Ontologies, including a general comparison to the domain-specific ones, discussion on their relations with the lexical knowledge bases and some incompatibility issues. Section 3 discusses the representation languages and primitives with subsections about the most significant paradigms and a short overview of the approaches for their unification. Section 4 describes the primitives used in the OntoMap project – the OntoMapO ontology. Section 5 focuses on the OntoMapO relations used for mapping concepts between ontologies. Section 6 enumerates the formats in which OntoMap will provide all the ontologies and mappings. The initial set of ontologies to be hosted and the mappings between them are discussed in section 7. Finally section 8 will give you an idea about the services that could be provided based on the results of the project.

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 "table" or "meeting". Most of the tables have 4 legs, however, there are pretty obvious exceptions - tables with three legs, a single leg or even without anything to be considered as a leg. There could also be a "serious" table with 6 legs. Then 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. That is why upper-level ontologies (for example, Upper Cyc Ontology, Sensus, MikroKosmos) defined mainly in terms of taxonomic relations. An attempt to strongly use attributes in their definitions could be hard, expensive, and usually leads to involvement of default reasoning or other similar mechanisms that cause intractability.

2.1 Upper-Level vs. Domain-Specific

This pseudo-dilemma seems to be mostly a question of goal and scope of the developers of the ontology rather than a representational or management problem. However, we want to outline that there exists a significant 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 example 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, thesauri, or dictionaries that attempt to formalize the lexical semantics — the meanings of the words in a natural language. 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, 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 an example is the SENSUS ontology. Other upper-level ontologies were developed in order to give formal semantics to a LKB - such example is the EuroWordnet Top Ontology, [12]. This is our reason to include a number of LKB semantic resources 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 goal OntoMap project is after, seems 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? For example, the top of the MikroKosmos ontology (see [11]) demonstrates a typical top-level:

```
ALL
    PROPERTY
        ATTRIBUTE
        RELATION
    OBJECT
        SOCIAL-OBJECT
        PHYSICAL-OBJECT
        MENTAL-OBJECT
        INTANGIBLE-OBJECT
    EVENT
        SOCIAL-EVENT
        PHYSICAL-EVENT
        MENTAL-EVENT
```

However, it ignores the stuff/object (countable) distinction that is very basic in Cyc (see [1]) and other upper-level ontologies.

Our understanding is that OntoMap should not try to choose the best upper-model or to produce a new one. The upper-level have to be chosen according to the specific application – we just want to make the comparison easier.

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 of the knowledge representation. Here is a non-exclusive overview of the most popular "languages" used by the ontologists:

- **concepts, properties** — these are usually the terms inspired by the early **semantic networks** (SemNet), mathematics and philosophy. *Concepts* are used to denote any kind of static and cognitively autonomous semantic phenomena. They classify the entities in the domain of discourse — each entity either belongs to a certain concept's interpretation, or not. In other words, the information carried out by the concept is either true for the entity either not. The entities that belong to the interpretation of the concept are called *instances* of the concept. Typical concepts are Person, Food, Meeting, Idea. Then *properties* come to represent characteristics, aspects, or attributes of the entities, as well, as relationsships between them. They are further separated into *attributes* and *relations*. Typical representatives are: color, gender (attributes); loves, causes (relations).

- **classes, slots, facets and frames** — obviously, this is the frame-based terminology (**Frames**). Here *classes* correspond to the concepts while the notion for the instances remains the same. *Slots* (especially as they evolved in the last years) correspond to the properties. They are further classified as *template-slots* (class- or concept-attributes in SemNet) and *own-slots* (instance-attributes in SemNet). The template-slots are defined on a class level — for example, Color is a template-slot for the class Car. In contrast, own-slots connect some values of the template-slots to certain instances of the class, say Colour(Ferarri, Red). *Facets* are properties of the slots, for example, Domain, Range, Min-Cardinality, Documentation. Formally speaking, they are own-slots of the slots. There is no clear favorite for a single term corresponding to the facet notion in the rest of the paradigms;

- **concepts, roles, individuals** — this is the terminology used in the so called description logics (DL), the descendants of the KL-One knowledge representation language (CLASSIC, LOOM, KRIS, FaCT). This paradigm is pretty close to the one used in the semantic networks. Strictly speaking, it is developed to make them more precise on epistemological level. The *roles* correspond to the properties, the *individuals* correspond to the instances;

- **classes, objects, attributes** - this is the terminology used in the **object-oriented** paradigm (OO), mostly developed for the purposes of the software engineering. The *classes* correspond to the concepts while the *attributes* (also called data members) correspond to properties. An equivalent of the class-attributes are the *static data members*. The *objects* are always instances of certain class;

- **collections, individuals, predicates, constants** - this is the terminology used by the *cyclists*, the people developing the Cyc knowledge base at Cyc Corp. Roughly, the *collections* correspond to concepts, *individuals* to the instances, and binary predicates (that are kind of collections themselves) correspond to properties. The *constants* are names of collections, individuals, or predicates.

3.2 The Conceptual Level

Many attempts have been made to resolve the terminology diversity by managing ontologies in a representation-independent fashion on the so called knowledge-level or conceptual level. Two such approaches are reported in [4] and [9]. Even sticking to the frame-based terminology the knowledge-model of Protégé-2000 (see [10]) 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 [7] — 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 [1] and the previous sub-section) 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 as fatal for the overall usability as a complex system of primitives could be. Thus we undertake an approach opposite to the one employed in Ontolingua, [5], following the rationale that even though many distinctions could be clearly defined there 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 some 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 [17] and [16]) s/he has to bother about the meaning of slots like `Set-Cardinality` and `Relation-Universe`.

We tried to develop 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. Another primary consideration was to keep it decidable.

Here is the point to mention that OntoMapO ontology could be also regarded 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 could perform equally well.

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 from Cyc ([1]), Protégé-2000 ([10]), and RDFS ([13]) representation models — in addition to the inheritance relations we also employ instantiation as a basic mechanism. 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. This way an infinite number of meta-levels can 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 $I \in C$. In the same fashion `(ChildOf C1 C2)` means that $C1 \subset C2$. 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 $C1 \subset C2$ and $C2 \subset C3$ it follows that $C1 \subset C3$
- the instantiation is not-transitive — if $I \in C$ and $C \in MetaC$ it does NOT follow that $I \in MetaC$
- the instantiation is transitive with respect to inheritance — if `(InstanceOf I C1)` and `(ChildOf C1 C2)` it follows that `(InstanceOf I C2)`. Really, from $I \in C1$ and $C1 \subset C2$ it follows that $I \in C2$

An obvious advantage of such extensive use of instantiation is that it makes the hierarchy less tangled avoiding multiple-inheritance in many cases. Let us take as an example `ChildOfInterO` relation that is explained in details later on in the paper. If we were not able to define it as transitive inter-ontology relation using instantiation it should have been defined as child of three parents caring different information: transitiveness, its inter-ontology nature, and the fact that it is a kind of inheritance relation. We preferred to encode the first two aspects by instantiation to the corresponding meta-concepts and define `ChildOfInterO` only as a child of the `ChildOf` relation that is most related to its nature and identity.

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)` than the pair $< B, C > \in RelA$. 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 us 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 $< \mathtt{A}, \mathtt{B} > \in$ `RelA`

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

- now it is obvious that $< \mathtt{A}, \mathtt{B} > \in$ `RelB`, which 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`. Analogically, 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 Each OntoMapO Relation has an Inverse Relation

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 twofold:

- 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

- two 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. Analogically, `Reifies` is inverse to the `Range` relation that holds between a relation type and a concept whose 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 How are the Predefined Relations Special

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 a sub-relation of `Equivalent`. In a similar fashion, all the sub-relations of `ChildOf` and `ParentOf` are treated as inheritance relations. Analogically, 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 `(EquivalentInterO #$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. First, after each concept name in brackets we have the classes it belongs to. Actually, only the informative classes are listed, i.e. the most specific ones. When a class have multiple parents (multiple-inheritance is allowed) it appears in the sub-tree of each of its parents.

```
Top (Concept)
   Concept (Concept)
      Range (BinaryRel)
      ClassOf (BinaryRel)
      Equivalent (TransitiveRel, SymmetricRel)
      ChildOf (TransitiveRel)
      Inverse (SymmetricRel)
      ParentOf (TransitiveRel)
      HasSlot (BinaryRel)
      Domain (BinaryRel)
      BinaryRel (Concept)
         TransitiveRel (Concept)
         InterOntoRel (Concept)
         SymmetricRel (Concept)
      InstanceOf (BinaryRel)
      Reifies (BinaryRel)
   Ontology (Concept)
      Module (Concept)
```

The above hierarchy does not include the ontology mapping primitives discussed in the next section.

5. ONTOLOGY-MAPPING PRIMITIVES

All the relations that can take place between concepts from different ontologies are represented in the OntoMapO as instances of `InterOntologyRel`. On the other hand many of them are defined as specifications of the corresponding base relations. For example, `ChildOfInterO` is nothing more than `ChildOf` relation between concepts from different ontologies. We are defining such relations as specification of the basic ones in order to allow easy distinction between the relations inside the ontologies and the mappings.

There are also inter-ontology relations that do not have a corresponding intra-ontology relations. The reason is that those should be used for handling structural differences between different ontologies. For example, the `(TopInstance-InterO A B)` should be used when a concept `A` from one ontology exists as a concept `B` on the next meta-level in another ontology, i.e. `(ChildOf X A)` holds iff `(InstanceOf X B)` holds. The point is that we think that such design patterns should not be tolerated inside a single ontology.

Here follows the list of all inter-ontology relations:

- *equivalent* – the first concept is equivalent to the second one, transitive

- *more specific* – the first concept is more specific than the second one, transitive. Inverse to *more general*

- *more general* – the first concept is more general than the second one, transitive. Inverse to *more specific*
- *much more specific* – the first concept is much more specific than the second one, transitive. Inverse to *much more general* and a specification of *more specific*
- *much more general* – the first concept is much more general than the second one, transitive. Inverse to *much more specific* and a specification of *more general*
- *top-instance* – the first concept is the most general instance of the second one, that is a meta-concept. Inverse to *exact meta-concept* and a specification of *instance*
- *exact meta-concept* – the first concept is a meta-concept, the second concept is the most general instance of the first one. Inverse to *top-instance* and a specification of *meta-concept*
- *instance* – the first concept is an instance of the second one, that is a meta-concept. Inverse to *meta-concept*
- *meta-concept* – the first concept is a meta-concept, the second concept is its instance. Inverse to *instance*
- *super-instance* – the first concept is more general than all the instances of the second one that is a meta-concept. Inverse to *sub-meta-concept*
- *sub-meta-concept* – the first concept is a meta-concepts, all its instances are more specific than the second concept. Inverse to *super-instance*
- *inverse* – both concepts are relations that are inverse to each other, symmetric

Here follows the part of the OntoMapO hierarchy that represents only the mapping primitives and their parents. All the relations which names end up with `InterO` are instances of `InterOntoRel` — in order to simplify the hierarchy this information will not be included in the hierarchy.

```
Top (Concept)
  Concept (Concept)
     ClassOf (BinaryRel)
        ClassOfInterO
           ExactClassOfInterO
     Equivalent (TransitiveRel, SymmetricRel)
        EquivalentInterO (SymmetricRel)
     SuperInstanceInterO
     ChildOf (TransitiveRel)
        ChildOfInterO (TransitiveRel)
           MuchMoreSpecificInterO (TransitiveRel)
     Inverse (SymmetricRel)
        InverseInterO (SymmetricRel)
     ParentOf (TransitiveRel)
        ParentOfInterO (TransitiveRel)
           MuchMoreGeneralInterO (TransitiveRel)
     SubMetaConceptInterO
     InstanceOf (BinaryRel)
        InstanceOfInterO
           TopInstanceOfInterO
```

5.1 Representing Ontologies in OntoMap

When an ontology representation has a well defined conceptualization our approach is to map its primitives to the OntoMapO primitives. For example, importing Upper-Cyc Model ([1]) we just defined that

```
(EquivalentInterO #$genls ChildOf)
(EquivalentInterO #$isa InstanceOf)
```

an so forth with the rest of the Cyc's relations. While OntoMapO interprets each relation that is equivalent or more specific than `ChildOf` as inheritance relation it starts perfectly understand the inheritance in Cyc.

By analogy importing Protégé-2000 ([10]) meta ontology we can establish that:

```
(EquivalentInterO :DIRECT-SUPERCLASSES ChildOf)
(EquivalentInterO :DIRECT-SUBCLASSES ParentOf)
(ChildOfInterO :DIRECT-INSTANCES InstanceOf)
(ChildOfInterO :DIRECT-TYPE ClassOf)
```

First, let us answer why `:DIRECT-TYPE` is more specific than `ClassOf` — in Protégé each concept could be instance just of a single class. This limitation makes `:DIRECT-TYPE` more specific relation than `ClassOf`. Even with this complication, OntoMap will be able to interpret the instantiation as it is defined in otégé because:

- `ChildOfInterO` is a specification (sub-relation) of `ChildOf`, so
- `ChildOfInterO` is an inheritance relation, so
- `:DIRECT-TYPE` is a specification (sub-relation) of `ClassOf`, so
- `:DIRECT-TYPE` is an instantiation relation.

6. UNIFORM ACCESS

OntoMap will provide a number of alternative and complementary access methods to the hosted ontologies and mappings. These could be split on three layers:

- pragmatic — such as FIPA, KQML, OKBC. At this level the pragmatic of the communication is defined. At least types of messages are defined that allow to distinguish between questions, answers, unsolicited information, etc.
- epistemological — such as KL-ONE, Conceptual Graphs, OntoMapO. At this level the semantic primitives used to express the content of the communication is specified.
- transport protocol — such as RMI, CORBA, SOAP, HTTP. The technical approach for transmission of the information is specified here

In many cases one standard covers more than one of these layers. For example, OKBC specifies both the pragmatic and the epistemological level. Our approach is to provide various options for all the layers.

6.1 FIPA Compliant Ontology Service Agent

The Foundation for Intelligent Physical Agents (FIPA) has put substantial effort in developing specifications for interoperability between agent based applications. As a result, development of agent based applications is easier and such applications are adopted by different industries easier.

A fundamental concept in the FIPA framework is the Agent Platform (AP) — a combination of hardware, non-agent software, agent support software and agents. Each AP has a single Agent Management System (AMS, see [14]) which controls the proper operation of AP and agent life-cycle. An AP has one or more Directory Facilitators (DF), which offer directory services (such as symbolic name resolution) for the agents in the AP. Optionally an AP may also include an Ontology Agent (OA) providing ontology services to the AP.

As described in [15], an OA offers services like:

- access to public ontologies
- translation of expressions between different ontologies
- response to queries about relationship between terms
- identification of a shared ontology that could be used for communication between two agents

Thus a FIPA compliant OA is a solution which will offer a standardized access (on pragmatic level) to the OntoMap.org functionality for FIPA compliant agent systems. Even more, the semantic framework of OntoMap could be seen as an extension of the OA specification at epistemological level — that will allow easier development of reusable ontology agents.

6.2 Formats and Representations

In addition to the ontology agent publicly available ontologies (or parts of them) will be presented in a number of standard forms:

- PROLOG
- KIF
- XML (according to OIL and according to OntoMapO)
- RDF(S)
- HTML - an online ontology browser as well as static pages available for download
- SQL scripts for ORACLE and MS SQL Server
- Ready-to-use files for MS Access (MDB)
- Online application server accessible via CORBA, EJB, RMI, and SOAP (an RPC protocol based on XML)

7. THE INITIAL SET OF ONTOLOGIES

The following ontologies will be initially hosted :

- Upper Cyc Ontology [UCYC]
- EuroWordnet Top Ontology [EWNTOP]
- EuroWordnet Clusters [EWNCLUST] - the clusters of EWN base concepts classified by top concepts. An extension of ETOP
- WordNet 1.5 unique beginners [WNUB5]
- WordNet 1.6 unique beginners [WNUB6]
- CORELEX [CLEX] - made on top of WNUB5
- SIMPLE Core Ontology [SIMC]
- 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 its creators will be available as well.

Mappings between some of the ontologies will be provided in order to ensure easier understanding and comparison between them. So, the ontologies covered will form an inter-connected graph. Such a mapping already exists between EWNTOP and UCYC (see [8]). The mapping between MKOSTOP and the latter two ontologies was recently developed within the project. Some of the relations in the graph exist because of the nature of the ontologies:

- EWNCLUST and ETOP - the concepts of the former one are just conjunctions of those of the later one
- CLEX and WNUB5 - same as above
- WNUB5 and WNUB6 - there exist a mapping provided by the creators
- SENSTOP and UCYC - it is available as a part of the UCYC distributives as well as separately by the SENSUS developers.

The following mappings will be developed as a part of the project:

- EWNCLUST to UCYC
- CLEX to EWNCLUST (both directions)
- ETOP to SIMC
- WNUB6 to UCYC
- WNUB6 to EWNTOP
- WNUB6 to SIMC

The mappings will be available in the same formats as the ontologies. Actually, each mapping could be seen as an extension of the target ontology. For example, the mapping between EWNTOP and UCYC can be considered as an extension of the UCYC with the concepts of EWNTOP that are connected appropriately to the UCYC constants (see [8]). No ontologies and upper-level models will be developed in the project instead of the OntoMapO meta-ontology mentioned above.

8. AN EXAMPLE

Here follows an example of how 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. This constant's representation in [1] 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: **#$MeetingTakingPlace** [UpperCyc]
>
> **Gloss:** The collection of human meeting events, in which #$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.
>
> **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], SituationComponent[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 attention that both indirect parents and classes are displayed which is pretty useful — it requires serious efforts to reconstruct this indirect relations manually. Also, super-concepts in the EuroWordnet (EWN) Top Ontology can be seen, which provide good impression about a possible position of `#$MeetingTakingPlace` there. These way people that are familiar with EWN top can get an idea about the meaning of the Cyc constant.

9. ONTOLOGY UNIFICATION SERVICES

After the end of the project, in parallel with the maintenance of the server we will be able to provide the following services for both domain specific and upper-level ontologies:

- Loading the ontology in OntoMap and hosting it there. This way it will become accessible in all the formats supported. Also its conformance profile could be determined (see Unified Representation section and [3]). A security subsystem will be developed, so the proprietary ontologies will not be publicly available.
- Developing of mappings to ontologies that are already hosted in OntoMap. The mappings themselves will be also available in all of the supported formats.

10. CONCLUSION

OntoMap project is still in an early phase that makes it hard to evaluate it. The experience gathered providing the Upper Cyc Ontology as MS Access database is encouraging – even though the resource is available for a long time some 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 development of the community.

The first practical result of the project is a Java-written inference engine that already 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 few seconds and then queried real-time.

Furthermore, our intent is that the OntoMap server is not limited only to upper-level ontologies. Later phases of the project will aim at providing the mapping services for domain specific ontologies - the ontology community will be supplied with the ability to create/upload new ontologies for the OntoMap server and define the mappings to ontologies already available there. The underlying semantic framework is most suitable for upper-level ontologies, but there are no principle problem to use it for domain specific ontologies. It have to be mentioned that in such cases the information loss (compared to the ontology in its original representation) will be bigger because such ontologies usually rely on much more powerful representation languages.

11. REFERENCES

[1] Cycorp, 1997. *Cyc Ontology Guide: Introduction.* http://www.cyc.com/cyc-2-1/intro-public.html

[2] Fensel, Dieter *Ontologies: Their Glory and the new bottlenecks they create.* Presentation on "Semantic Web Project Proposal" workshop, Vrije Universiteit Amsterdam (the Netherlands), Dec 8, 2000 `http://www.ontoweb.org/workshop/ amsterdamdec8/`

[3] Genesereth, Michael R., and Fikes, Richard eds. *Knowledge Interchange Format draft proposed American National Standard (dpANS).* NCITS.T2/98-004 http://logic.stanford.edu/kif/

[4] 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

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

[6] Gruber, Thomas R. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* Technical Report KSL 93-04, Knowledge System Laboratory, Stanford University, 1993.

[7] Guarino, Nicola; Welty, Christopher *A Formal Ontology of Properties.* In the Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France. R. Dieg and O. Corby (Eds.): EKAW 2000, LNAI 1937, pp. 97-112, Springer Verlag, 2000.

[8] Kiryakov, Atanas; Simov, Kiril Iv. *Mapping of EuroWordnet Top Ontology to Upper Cyc Ontology.* In: Proceedings of "Ontologies and Text" workshop, during EKAW 2000. Juan-les-Pins, French Riviera, Oct. 2, 2000. `http://www.ontotext.com/publications/ index.html# KiryakovSimov2000b`

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

[10] Noy, Natalya F.; Fergerson, Ray W.; Musen, Mark A. *The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility.* In the Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France. R. Dieg and O. Corby (Eds.): EKAW 2000, LNAI 1937, pp. 97-112, Springer Verlag, 2000.

[11] Ortiz, Antonio Moreno. *Managing conceptual and terminological information in a user-friendly environment.* In: Proceedings of "OntoLex 2000: Ontologies and Lexical Knowledge Bases", Sozopol, Sept. 8-10, 2000. (to appear)

[12] Vossen, Piek (ed.) *EuroWordNet General Document Version 3, Final*, July 19, 1999. `http://www.hum.uva.nl/ ewn/`

[13] 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/`

[14] The Foundation for Intelligent Physical Agents *FIPA Agent Management Specification*, November 30, 2000. `http://www.fipa.org`

[15] The Foundation for Intelligent Physical Agents *FIPA Ontology Service Specification*, June 15, 2000. `http://www.fipa.org`

[16] Uschold, Mike; King, Martin; Moralee, Stuart; and Zorgios, Yannis *The Enterprise Ontology*, The Knowledge Engineering Review, 1998, Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).

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