Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing

A. Gómez Pérez

M. Gruninger

H. Stuckenschmidt

M. Uschold

(*editors*)

Held on August 4 and 5, 2001 in conjunction with the
International Joint Conference on Artificial Intelligence
Seattle, USA

IJCAI-01 Workshop on
Ontologies and Information Sharing

Held on August 4 and 5, 2001 in conjunction with the

International Joint Conference on Artificial Intelligence

Seattle, USA

From the early 1990s, there has been a fruitful series of over a dozen workshops, symposiums and conferences on the emerging field concerned with the development and application of ontologies. Early workshops were focused in large part on identifying what ontologies were, and how they *might* be used. As the field developed and matured, we have obtained a reasonable understanding and consensus about the nature of ontologies. The core idea is to explicitly encode a shared understanding of some domain that can be agreed among different parties (be they people or computers). This shared understanding is the ontology – it is an explicit representation comprising a vocabulary of terms, each with a definition specifying its meaning. All parties commit to using these terms in accordance to their definitions.

Although there was a consensus on what an ontology *was*, how exactly ontologies could be created, put to use, and what enabling technologies would be required remained unclear. Thus, subsequent workshops focused mainly on the theoretical aspects of engineering, designing, building, maintaining and applying ontologies. There were a number of tools and methodologies were developed to assist building ontologies. However, the methodologies being reported were either immature or were developed and tested in one domain only. Also, there were few if any practical applications being reported, other than immature research prototypes.

In the last two years the aim of ontology workshops has shifted towards promoting a deep understanding of how ontologies may be applied in *working systems*. Frameworks for understanding ontology representation languages, ontology development environments, and ontology applications have been described. Also, a growing number of applications have been reported in a wide variety of areas. These include: e-commerce, knowledge management, enterprise modelling, intelligent integration information, communication between people and organisations, knowledge discovery in databases and data-mining, interoperability between systems (data bases, digital libraries), knowledge elicitation from text and the web, etc. There has also been some significant, theoretically well-founded work in providing methodological assistance for constructing ontologies.

The goals of the current workshop are twofold:
1. to go into detail in one particular application area: information sharing
2. to continue the further understanding of the field in general from both theoretical and practical standpoints.

Although the ways that ontologies may be applied are many and varied, a strong recurring theme has always been on *information sharing*. The central problem is the heterogeneity of data, information and knowledge that different people and computers need access to.

In the past, several approaches have been developed to reconcile the heterogeneity of data structures (e.g. federated databases or existing middleware solutions). Very often, these approaches are not able to satisfy all needs for the integration of data. Therefore, *semantic approaches* considering the intended meaning of terms in a special context or application must be developed. Theoretical and application-oriented approaches that are developed to achieve semantic interoperability are vital and therefore welcome. Scientists will have the opportunity to discuss new developments, innovative implementations and new ideas.

To further our understanding of this key application area, we aim to discuss state-of-the-art technologies, identify open problems and outline a further program of research to progress our understanding and reap the benefits of applications in the area of information sharing. To make this a success, we aim to attract researchers in a variety of application areas concerned with information sharing, to complement the core ontology researchers.

We received 23 submissions for the workshop. Of these submissions 18 were selected for oral presentation at the workshop. These proceedings contain long and short papers on the topics presented at the workshop. These topics can roughly be categorized into five main topics of interest

Foundations and Languages

While applications become more and more important, there are still open questions concerning the nature of ontologies and the way they should be encoded in order to be useful. **Gangemi and others** present a methodology for selecting top-level ontological categories and introduce domain independent relations for analyzing these categories. **Euzenat** addresses the problem of handling ontologies that have been encoded in different languages. He reviews and compares existing approaches for solving this problem using a model-theoretic framework. The contribution of **Stuckenschmidt** focuses on the idea of customizing ontology languages for specific applications and proposes a general approaches that is in line with the ideas of Euzenat**. Wohner** uses the term 'application semantics' to refer to the problem of using ontologies for a specific application. He discusses ontology 'laws', i.e. special properties of ontologies and discusses their potential impact on an application. **Tamma and Bench-Capon** finally present a formal model to describe meta properties of concepts including which properties are prototypical of a concept and which are exceptional, the behaviour of properties over time and the degree of applicability of properties to subconcepts.

Ontology Engineering

A well-known problem connected with the use of ontologies is the acquisition and formalization of generalized knowledge. Several papers address this problem from different points of view. **Boicu and others** demonstrate, how reuse of existing ontologies together with translation and machine leraning techniques can be used to ease and speed up the knowledge acquisition process. Their approach is demonstrated using an example from the DARPA RKF programme. **Stevens and others** describe their experiences with using the OIL language and associated tools to build a bioinformatics ontology. Based on a case study they emphasize the assistance provided by the description logic based reasoning service that can be used to structure the ontology. The work decsribed by **Golebiowska** and others is concerned with a different application domain, namely the managemnet of knowledge in complex development processes. They describe techniques applied in a automobile project and draw some conclusions about the use of ontologies for knowledge management. **Gandon** discusses the use of ontologies in multi-agent information systems focussing on the development process and the use of standards.

Ontology Integration

Quite a number of papers address the problem of integrating different ontologies. The paper of **Klein** contains a classification of different tasks and problems related to the combination of different ontologies. Klein further reviews some existing approaches to ontology integration. Two of the submitted papers contain concrete approaches for ontology integration: **Noy and Musen** present Anchor-PROMPT, an extension of their previously developped system PROMPT. The extension tries to exploit the overall structure of an ontology in order to determine similar classes. **Stumme and Mädche** present FCA-Merge, an approach that combines techniques from natural language processing and formal concept analysis to derive a lattice of concepts, starting with instances from two ontologies as input. While these contributions focus on algorithms and systems, the work of **Sofia-Pinto and Martins** try to develop guidelines for the process of integaring ontologies, thus supplementing general development methodologies.

Applications of Ontologies

Today, ontologies are understood well enough to be used in real-life applications. Quite a number of contributions report or review successful applications of ontologies. **Wache and others** review existing work concerned with the use of ontologies for information integration. Their survey of over twenty existing systems shows that information integration is an interesting application area where significant results have been achieved. However the paper still points out to open problems worth being discussed. **Kalfoglou and Vargas-Vera** present a special module of OntoWeb – an intelligent news broadcast system – that allows the personalization of information services based on ontologies.

Ontologies and E-Business Applications

In a joint session with the IJCAI-01 Workshop on E-Business and the Intelligent Web, Applications of ontologies in the E-Business domain are presented. Three contributions were selected for presentation in this special session. **Corcho and Gomez-Perez** demonstrate the use of multi-lingual ontologies for the integration of e-commerce standards used in B2B marketplaces**. Flett and Brown** report on the activities at SemanticEdge that are undertaken to develop standardized ontology tools for real-life applications. **Izumi and Yamaguchi** finally propose an ambitious methodology for building business applications based on explicit business models that are integrated using ontologies. The joint session is completed by three papers that have been submitted to the E-Business and the Intelligent Web Workshop. These papers are also included in the Proceedings.

Program Committee and Reviewers.

We are grateful to the following members of the international program committee for helping us to make this a high quality workshop:

- Mike Brown, SemanticEdge, Berlin, Germany
- Jerome Euzenat, INRIA Rhone-Alpes, France
- Dieter Fensel, Free University Amsterdam, The Netherlands
- Nicola Guarino, LADSEB-CNR, Padova, Italy
- Frank van Harmelen, AI Department, Free University Amsterdam, The Netherlands
- James Hendler, DARPA, USA
- Ora Lassila, Nokia Research Center, Boston, USA
- Deborah McGuiness, KSL, Stanford University, USA
- Enrico Motta, KMI, Open University, UK
- Mark Musen, Stanford University, USA
- Christoph Schlieder, Center for Computing Technologies, University of Bremen, Germany
- Steffen Staab, AIfB, University of Karlsruhe, Germany
- Rudi Studer, AIfB, University of Karlsruhe, Germany
- Gerd Stumme, AifB, University of Karlsruhe, Germany
- Ubbo Visser, Center for Computing Technologies, University of Bremen, Germany
- Holger Wache, Center for Computing Technologies, University of Bremen, Germany

We would also like to thank the following additional reviewers:

- Oscar Corcho, Facultad de Informática, Universidad Politécnica de Madrid, Spain
- Mariano Fernandez-Lopez, Facultad de Informática, Universidad Politécnica de Madrid, Spain
- Yannis Kalfoglou, KMI, Open University, UK
- Alexander Mädche, AIfB, University of Karlsruhe, Germany
- Claudio Masolo, Dept. of Electronics and Computer Science, University of Padova, Italy
- Alessandro Oltramari, LADSEB-CNR, Padova, Italy
- Ingo Timm, Center for Computing Technologies, University of Bremen. Germany
- Maria Vargas-Vera, KMI, Open University, UK

Related Workshops of the last years

This Workshop continues the series of ontology-related workshops listed below, in which members of the organization committee were strongly involved. It thereby adopts ideas and questions, which have been discussed at various workshops on information integration and sharing also listed below

Workshops on Ontologies:

- Applications of Ontologies and Problem-Solving Methods, ECAI 2000 http://delicias.dia.fi.upm.es/WORKSHOP/ECAI00/index.html
- Ontology Management, AAAI 1999, Orlando, Florida, USA. http://www.aaai.org/Workshops/1999/ws-99.html
- Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, IJCAI'99 http://www.swi.psy.uva.nl/usr/richard/workshops/ijcai99/home.html
- Formal Ontologies in Information Systems, FOIS-98, Italy http://www.ladseb.pd.cnr.it/infor/ontology/FOIS98/FOIS98.html
- Cost Effective Development and use of Ontologies and Problem Solving Methods, KAW'99 http://sern.ucalgary.ca/KSI/KAW/KAW99/
- Applications of Ontologies and Problem-Solving Methods, ECAI'98 http://delicias.dia.fi.upm.es/WORKSHOP/ECAI98/index.html
- Shareable and Reusable Components for Knowledge Systems, KAW'98 http://spuds.cpsc.ucalgary.ca/KAW/KAW98/KAW98Call.html

Workshops on Information Sharing and Integration:

- Workshop “Information Sharing” at the International Symposium on Computer Science for Environmental Protection (UI 2000) http://www.giub.uni-bonn.de/ui2000/cfp_is.html
- Third Workshop on Intelligent Information Integration at the International Joint Conference on Artificial Intelligence (IJCAI-99) http://www.aifb.uni-karlsruhe.de/WBS/dfe/iii99.html
- Second International Workshop on Practical Information Mediation, Brokering, and Commerce on the Internet (I'MEDIAT'99), 1999 http://context.mit.edu/imediat99
- First International Workshop on Information Integration and Web-based Applications & Services (IIWAS'99), 1999 http://www.te.ugm.ac.id/iiwas99.html
- Second Workshop on Intelligent Information Integration at the European Conference on Artificial Intelligence (ECAI-98) http://www.tzi.de/grp/i3/ws-ecai98/
- First International Workshop on Practical Information Mediation, Brokering, and Commerce on the Internet (I'MEDIAT'98), 1998 http://context.mit.edu/imediat98

There have been a couple of other workshops on ontologies before 1998. Prominent examples are corresponding workshops at ECAI 96 and IJCAI 95. Beyond this, ontologies have been discussed as an important enabling technology at workshops not directly dedicated to ontologies. Examples are the Dagstuhl seminar ‘Semantics for the Web’ in March 2000. We can also mention several workshops on Knowledge Management.

Table of Contents

SHORT PAPERS AND STATEMENTS

JOINT SESSION WITH THE E-BUSINESS AND THE INTELLENGENT WEB WORKSHOP

Full Papers

Ontologies and the Knowledge Acquisition Bottleneck

Mihai Boicu, Gheorghe Tecuci, Bogdan Stanescu, Gabriel C. Balan and Elena Popovici
Learning Agents Laboratory, Department of Computer Science, MS 4A5
George Mason University, 4400 University Drive, Fairfax, VA 22030-4444
{mboicu, tecuci, bstanesc, gbalan, epopovic}@gmu.edu, http://lalab.gmu.edu

Abstract

Ontologies and information sharing have a major role to play in the development of knowledge-based agents and the overcome of the knowledge acquisition bottleneck. This paper supports this claim by presenting an approach to ontology specification, import, and development that is part of Disciple-RKF. Disciple-RKF is a theory, methodology, and learning agent shell for the rapid development of knowledge-based agents by subject matter experts, with limited assistance from knowledge engineers. The Disciple approach has been subject of intensive evaluations, as part of DARPA's "High Performance Knowledge Bases" and "Rapid Knowledge Formation" programs, demonstrating very good results.

1 Introduction

Ontologies and information sharing have a major role to play in the development of knowledge-based agents and the overcome of the knowledge acquisition bottleneck [Buchanan and Wilkins, 1993]. Indeed, building a knowledge base is too difficult a task to always start from scratch when a new knowledge-based system needs to be created. It makes more sense to reuse knowledge from related knowledge bases than to recreate such knowledge because this process should, in principle, be easier. Moreover, this reuse should also facilitate the communication between the systems because of their shared knowledge.

However, knowledge sharing and reuse are in themselves very complicated processes, especially if the systems involved have not been specifically designed for this purpose. How to design a knowledge-based system to facilitate knowledge sharing or reuse is an open research question.

In this paper we present an approach to rapid development of knowledge-based agents that illustrates several general methods and ideas related to ontology reuse and development. This approach is implemented in the Disciple-RKF learning agent shell.

Disciple-RKF is a tool for the development of a knowledge-based agent directly by a subject matter expert, with limited assistance from a knowledge engineer. Disciple-RKF contains a general problem solving engine, a learning engine and an initially empty knowledge base. The process of developing a Disciple agent for a specific application relies on importing ontologies from existing repositories of knowledge, and on teaching Disciple how to perform various tasks, in a way that resembles how an expert would teach a human apprentice when solving problems in cooperation. While teaching Disciple how to solve problems is a major feature of this system, in this paper we concentrate on its ontology-related aspects.

The next section describes the architecture of the Disciple-RKF shell. An important feature of this architecture is the structuring of the knowledge base into a general object ontology that can be imported and a set of problem solving methods or rules that can be learned from a subject matter expert.

Section 3 presents the general domain modeling methodology used with the Disciple approach. A characteristic feature of this methodology is that it produces an initial specification of the object ontology needed for the application knowledge base being developed. This ontology specification is the input to the ontology import module that is described in section 4. This module implements a general approach to ontology import.

Section 5 discusses several intelligent assistants that help in the complex process of extending and improving the object ontology. Then section 6 presents a practical approach for eliciting instances from subject matter experts, to populate the object ontology.

Section 7 discusses briefly the process of agent teaching and rule learning. This is continued in section 8 with a discussion of the ontology learning issue.

The knowledge base developed through the processes mentioned above can also be exported into existing knowledge servers, for further reuse. The knowledge export method of the Disciple approach is presented in section 9.

The work reported here has been done as part of the DARPA's High Performance Knowledge Bases program [Cohen *et al.*, 1998], and continues as part of the Rapid Knowledge Formation program [Burke, 1999]. These programs included intensive experimentation periods that tested the claim that with the latest AI technology knowledge bases can be built quickly and efficiently. The tests required the development of knowledge-based systems for solving several challenge problems, including the following ones: 1) the workaround challenge problem:

planning the repair of damaged bridges and roads [Jones, 1998; Tecuci *et al.*, 2000a]; 2) the COA challenge problem: critiquing military courses of action [Jones, 1999; Tecuci *et al.*, 2000b], and 3) the COG challenge problem: identifying strategic center of gravity candidates in military conflicts [Gilles *et al.*, 1996]. In section 10 we present experimental results from these evaluations that support the claims made in this paper.

We conclude the paper with a discussion of our future research related to ontology and information sharing.

2 The Disciple-RKF Learning Agent

Disciple-RKF contains a domain modeling and problem-solving engine that is based on the general problem (or task) reduction paradigm of problem solving, and is therefore applicable to a wide range of domains. In this paradigm, a problem to be solved (or a task to be performed) is successively reduced to simpler problems until the problems are simple enough to be immediately solved. Their solutions are then successively combined to produce the solution to the initial problem.

An important feature of Disciple-RKF is the structuring of the knowledge base into two distinct components: an object ontology and a set of reduction and composition rules. The object ontology is a hierarchical representation of the objects and types of objects from a particular domain, such as military or medicine. That is, it represents the different kinds of objects, the properties of each object, and the relationships existing between objects. The object ontology provides a representation vocabulary that is used in the description of the reduction and composition rules. Each reduction rule is an IF-THEN structure that expresses the conditions under which a problem (or task) P_1 can be reduced to the simpler problems (tasks) P_{11}, … , P_{1n}. Similarly, a composition rule is an IF-THEN structure that expresses the conditions under which the solutions S_{11}, … , S_{1n} of the problems (tasks) P_{11}, … , P_{1n} can be combined into a solution S_1 of P_1.

Dividing the knowledge base into an object ontology and a set of rules is very important because it clearly separates the most general part of it (the object ontology), from its most specific part (the rules). Indeed, an object ontology is characteristic to an entire domain. In the military domain, for instance, the object ontology will include descriptions of military units and of military equipment. These descriptions are most likely needed in almost any specific military application. Because building the object ontology is a very complex task, it makes sense to reuse these descriptions when developing a knowledge base for another military application, rather than starting from scratch. In the case of Disciple-RKF the ontology reuse is further facilitated by the fact that the objects and the features are represented as frames, based on the knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol. OKBC has been developed as a standard for accessing knowledge bases stored in different frame representation systems [Chaudhri *et al.*, 1998]. Therefore, importing an ontology from an OKBC compliant knowledge server, such as Loom [MacGregor, 1999], Ontolingua [Farquhar *et al.*, 1996], and Protégé [Fridman *et al.*, 2000] does not raise translation problems.

The rules from the knowledge base are much more specific than the object ontology. Consider, for instance, two agents in the military domain, one that critiques courses of action with respect to the principles of war, and another that plans the repair of damaged bridges or roads. While both agents need to reason with military units and military equipment, their reasoning rules are very different, being specific not only to their particular application (critiquing vs planning), but also to the subject matter experts whose expertise they encode.

3 Domain Modeling and Problem Solving

Domain modeling is the first and the most difficult activity when developing a knowledge base. First, the subject matter expert and the knowledge engineer have to develop a model of the application domain that will make explicit, at a qualitative and informal level, the way the subject matter expert performs tasks. In the case of Disciple-RKF this means modeling the process of performing a specific task as a sequence of qualitative and informal task reduction and composition steps. The knowledge engineer and the subject matter expert will consider a set of specific tasks that are representative of the set of tasks that the final agent should be able to perform. Then, for each of these tasks, they will represent the problem solving process as a sequence of task reductions (and, possibly, task composition) steps.

The left hand side of Figure 1, for instance, represents an example of task reduction modeling from the Course of Action critiquing domain. The task to perform is “Assess COA411 with respect to the Principle of Surprise”. To perform this assessment, the expert needs a certain amount of information about COA411. This information is obtained through a series of questions and answers that help reduce the initial assessment task to simpler and better-defined ones, until the expert has enough information to perform the assessment: "Report strength in surprise for COA411 because of countering enemy recon."

A main result of this modeling process is that it identifies the concepts and the features that need to be part of the object ontology in order for the agent to perform the type of reasoning illustrated in Figure 1. Indeed, the reasoning steps from the left hand side of Figure 1 reveal the need for the concepts and the features from the right hand side of Figure 1. The collection of all these concepts and feature represent a specification of the ontology that will have to be developed. In our approach, this specification guides the import of relevant ontological knowledge from external repositories such as CYC [Lenat, 1995], Loom [MacGregor, 1999], or Ontolingua [Farquhar *et al.*, 1996], as will be presented in the next section.

A second result of the modeling process are the task reduction steps themselves. They represent problem solving examples from which the Disciple agent will learn general

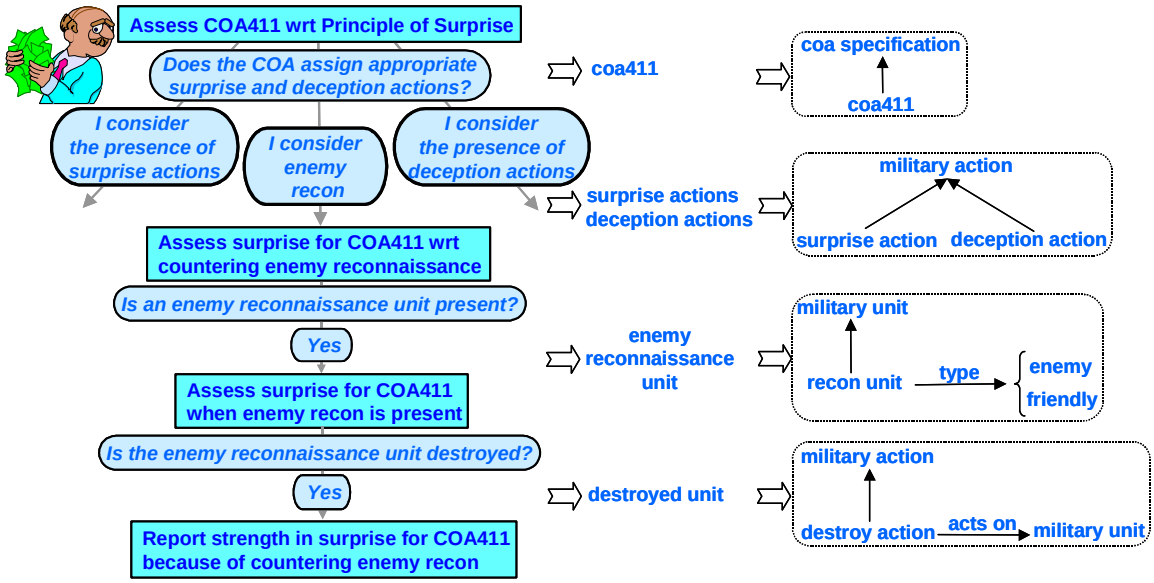


Figure 1: An illustration of the Disciple modeling process in the COA domain.

rules through the application of a mixed-initiative multistrategy learning method [Boicu *et al.*, 2000].

4 Ontology Import

As presented in the previous section, when the knowledge engineer works with the subject matter expert to define an initial domain model, they also identify the type of objects and features that are needed in the knowledge base (see Figure 1). These objects and features will focus the process of importing relevant ontological knowledge from existing knowledge repositories. The architecture of the ontology import module of Disciple is represented in Figure 2. Basically there are three phases of the ontology import process: 1) mixed-initiative retrieval of potentially relevant ontological knowledge from an external knowledge repository; 2) automatic translation of the retrieved ontological knowledge into an intermediate Disciple ontology; and 3) mixed-initiative import from the intermediate Disciple ontology into the final Disciple ontology. Each of these phases is discussed bellow.

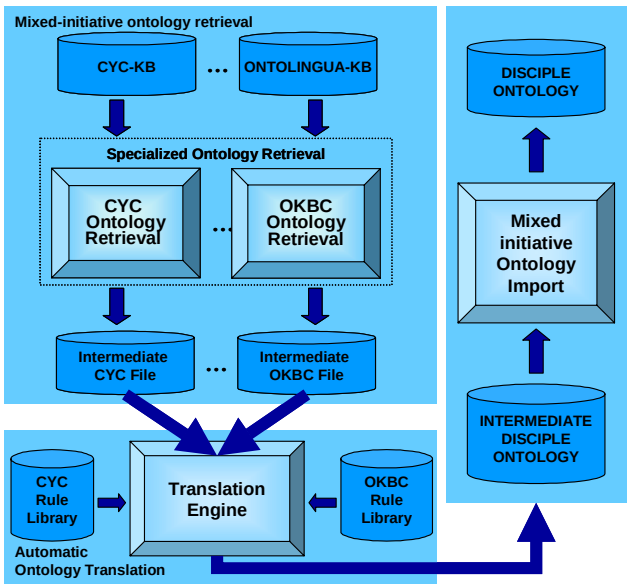


Figure 2: The Ontology import module.

In general, one of the practical difficulties encountered in ontology import is the fact that the subject matter expert has to deal with the additional representation system and tools of the knowledge repository from where knowledge has to be imported. To alleviate this problem, for each knowledge repository from which we are importing knowledge in Disciple, a standard ontology retrieval interface is developed. This interface allows the subject matter expert to retrieve relevant knowledge from different representation systems without dealing with the tools or representation of that knowledge repository. In the current Disciple architecture there are three planned implementations of the standard interface, one for the CYC system, which already exists, another one for any OKBC-compliant knowledge repository, such as Loom [MacGregor, 1999], Ontolingua [Farquhar *et al.*, 1996], or Protégé [Fridman *et al.*, 2000], and another one for older Disciple repositories.

Figure 3 illustrates the process of mixed-initiative retrieval of relevant ontological knowledge from the CYC knowledge repository. The subject matter expert introduces one of the terms needed in the ontology to be developed. A specialized CYC-searching module retrieves CYC terms that are likely to correspond to the input term, together with their documentation and pretty-names. Then the subject matter expert selects from the retrieved terms those that actually correspond semantically to the input term. This process is repeated for all the terms identified as relevant during domain modeling and results into a set of relevant

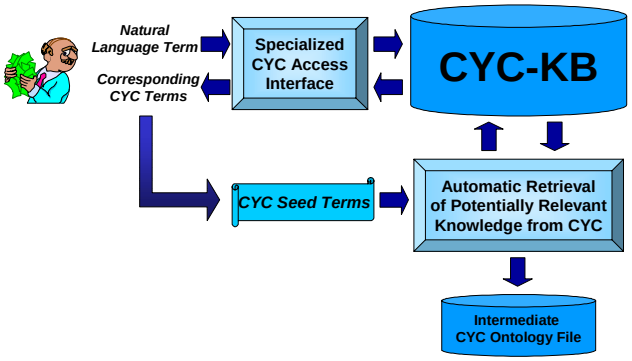


Figure 3: Mixed-initiative retrieval of relevant ontological knowledge

CYC terms, called seed. This seed represents the input to an automatic retrieval process that extracts from CYC all the terms that are related to those in the given seed. The automatic retrieval process is based on a breath-first search in a graph where the nodes are the terms and the edges are the CYC axioms that connect them. The result of this process is the transitive closure of the knowledge related to the seed, or a subset of it (the user has the possibility to specify a bound on the depth of the search or to stop the process at any time). The output of this process is a subset of the CYC ontology that is potentially relevant for the Disciple ontology to be developed.

In the second phase of the ontology import process, the retrieved CYC ontology is automatically translated into an intermediate Disciple ontology by a general rule-based translation engine that uses a CYC-Disciple rule translation library. Additional rule translation libraries need to be defined for each type of knowledge repository (e.g. for an OKBC-compliant knowledge server, for older Disciple repositories, etc.). Although we are currently using hand-written libraries of rules, we plan to use Disciple to learn general translation rules from the specific examples.

One important issue in ontology translation is the relative expressive power of the languages between which the translation takes place (see [Corcho and Gomez-Perez, 2000] for a comparison of the expressiveness of several ontology specification languages). As mentioned above, the representation of the Disciple object ontology is based on the OKBC knowledge model and is usually less powerful than the representations of the knowledge servers from which we need to import knowledge. On the other hand, the purpose of ontology import in Disciple is not to import the entire knowledge from the knowledge repository, but only the relevant knowledge that can be represented in the Disciple object ontology. This is because the primary purpose of the Disciple object ontology is to serve as a generalization hierarchy for learning of problem solving rules. Most of the representational and inferential power of Disciple does not come from the object ontology, but from the learned rules which we consider to be much more domain-specific and even expert-specific, and therefore less reusable and less likely to require importing.

The result of this translation process is an intermediate Disciple ontology which is the input for the third phase of the ontology import process. This intermediate ontology contains all the ontological knowledge retrieved from CYC or another knowledge repository. This is generally a very large ontology and only a relatively small part of it is likely to be useful for the final Disciple ontology to be built. The actual import is therefore taking place from this intermediate ontology. However, this is a Disciple ontology, and can be browsed using the Disciple tools. Therefore, the subject matter expert and the Disciple agent can collaborate to effectively import from it into the agent's ontology the object concepts that are considered useful.

An important feature of the Disciple approach is that most of the ontology import task can be done by the subject matter expert and the agent, with only limited assistance from the knowledge engineer. Also, the subject matter expert does not need to deal with the representation or tools of the external knowledge repository, but only with the representation of the system to be built (which, in this case, is Disciple). Finally, to be able to import knowledge from a new knowledge repository, the knowledge engineer would only need to implement a retrieval interface like the one in Figure 3, and to define rules to translate knowledge from the external repository to Disciple. These components are not very complicated. All the other components needed are independent of the external knowledge server.

5 Ontology Development

The imported ontology will generally need to be further extended and maintained. Disciple-RKF contains a set of browsers and viewers for easy navigation and visualization of the ontology. They include hierarchical browsers that allow the subject matter expert to navigate the ontology along the generalization relationships between the object concepts or the object features. There is also an association browser that allows the visualization of the object ontology as a network where the objects are the nodes and their relationships are the links. Navigating through this network is done by simply clicking on a object which becomes the center of the screen.

While visualizing and navigating the ontology are relatively simple tasks for a subject matter expert, modifying the ontology is a very complex task. For instance, let us consider the case where the user wishes to delete the subclass-of (is-a) relation between the concept B and the concept A (see Figure 4). This operation will not generate any inconsistency related to either A or B, but will generate an inconsistency for the sub-concept C of B. The concept C has the feature f, and this feature has the domain A (the domain of a feature represents the set of all objects that may have that feature). After removing B as sub-concept of A, the concept C will no longer be in the domain A of f, and therefore C may no longer have the feature f. As this example illustrates, a modification in one part of the ontology may generate subtle inconsistencies in other parts, and this makes ontology modification a very complex process.

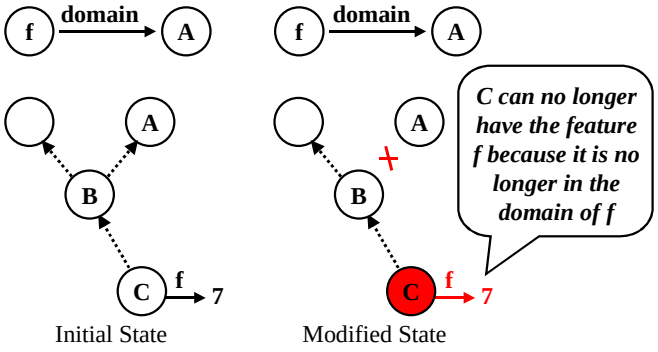


Figure 4: Inconsistency generated by a modification in the object ontology

In principle, there are two different approaches to ontology modification. The first one is to allow the user to introduce inconsistencies in the knowledge base and then to correct them. This approach is used in the Chimaera system [McGuinness *et al.*, 2000]. In this approach the modification of the knowledge base becomes an easy process. However, removing the inconsistencies is a very difficult process, which we think to be well beyond the capabilities that can be expected from a subject matter expert. Therefore we did not adopt this approach in Disciple. Instead, we adopted an approach where specialized ontology management assistants (which implement knowledge engineering methods and operations) guide and support the user in modifying the knowledge base such that the ontology will always be in a consistent state. There are assistants to create object concepts and features, to change the superconcepts of an object concept, to specify the value of a feature, to delete objects and features, to rename or copy them, and others. To implement these assistants we are developing a hierarchy of errors and warnings, as well as corresponding error correction methods.

The assistants operate according to the following scenario. The user formulates a goal, for instance to delete a given object concept. Then the corresponding assistant, which in this case is the delete assistant, analyses the knowledge base to determine all the implications of the operation intended by the user. It then notifies the user on the consequences of his or her planned action. After that a mixed initiative process is started to achieve the user's goal without introducing inconsistencies in the knowledge base. The assistant will propose specific knowledge management operations and the user may select the operations and guide the assistant to perform them.

6 The Input Ontology

After the object ontology is created, the agent can be trained to solve problems, as will be briefly presented in section 7. For this, one has to represent a problem in the agent's knowledge base. The part of the object ontology that is used to describe an input problem represents the input ontology. Let us consider, for instance, the most recent application of Disciple: identification of strategic center of gravity candidates in military conflicts. In 1832 Clausewitz introduced the concept of a center of gravity of a force as "the hub of all power and movement, on which everything depends" [Gilles *et al.*, 1996]. In this domain, an input problem is a description of a conflict scenario, such as the World War II planned invasion of Okinawa by the Allied forces in 1945. This includes the specification of the goals of the opposing forces, of the relevant factors (such as economic and geographical factors), and of the dominant factors (such as the composition of forces, the controlling and governing elements, and the type of civilization). Only after all this information is provided can Disciple reason about the potential centers of gravity of the opposing forces. From an ontology point of view, specifying an input problem (a scenario) consists of defining instances of the concepts from the input ontology, together with their features. This is not a trivial task for a subject matter expert. Therefore specialized elicitation forms are used to facilitate it, such as those from the Protégé system. For Disciple, we have developed a Scenario Elicitation module that allows the subject matter expert to create and update a scenario using a simple interface, which is illustrated in figure 5. The left hand side of the interface is a tree of titles and subtitles, similar to a table of contents. Each title (or node) corresponds to a certain type of information. When the expert clicks on such a node, Disciple requests relevant information about that node in the right hand side of the screen. If the expert has previously provided this information he can review or update it. The subject matter expert can go to any entry in this table of contents, to provide or update the information corresponding to that entry. Some information provided by the expert may lead to the creation of additional nodes in the left hand side of the interface. For instance, when the expert defined Japan-1943 and US-1943 as opposing forces, several nodes have been introduced in the left hand side of the interface.

The main idea of the implementation of the scenario elicitation module is to associate elicitation scripts with the concepts from the input ontology. The script associated with a concept plays multiple roles: it specifies how an instance of that concept is created; what features of the instance need to be elicited; how the dialog with the user takes place, and what graphical components are used in this dialog. In the current version of Disciple these scripts have to be developed by a knowledge engineer after the input ontology has been created. A single concept from the ontology is also marked as the starting concept for the scenario elicitation. In the example from figure 5, the starting concept is "Scenario". The right part of figure 5 shows the dialog between the system and the user. Once the user introduced the name of the scenario ("Okinawa"), the system created an instance of the "Scenario" concept. Then the script to elicit the features of the Okinawa scenario was activated. To elicit a specific feature, Disciple also uses the information from the ontology about that feature, such as the possible values and its cardinality. When the subject matter expert specifies a value of a feature that is an instance of some other concept, the script of that concept is activated and a new

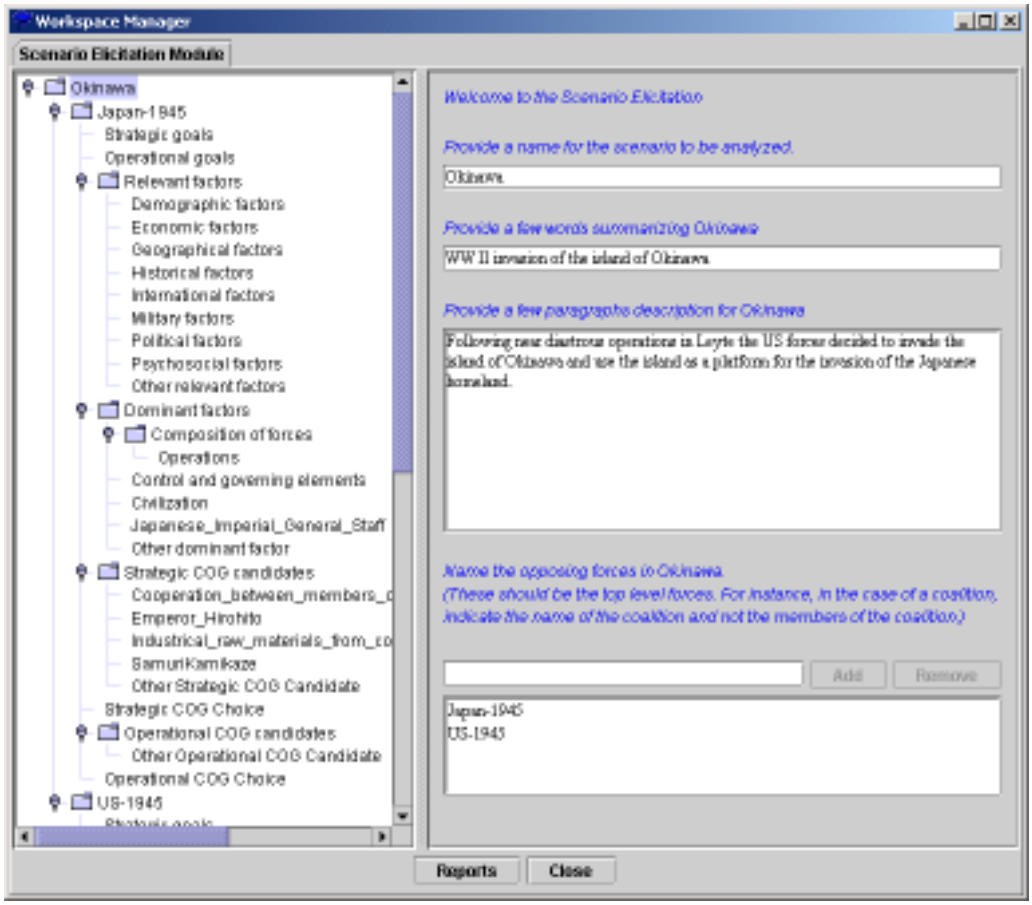


Figure 5: An interface for scenario elicitation

entry is added to the table of contents for the features of that instance. Figure 6 shows a fragment of the elicitation script for the concept "Scenario." This script requires the user to specify the opposing forces as illustrated at the bottom right of figure 5. Figure 7 shows the corresponding instances and relationships that have been introduced in the ontology.

Elicitation Script for the instances of the concept: *Scenario*
. . .
Property: *detailed-name*
Prompt: "Provide a few words summarizing " <*current-instance*>
Control-type: *single-line*
. . .
Property: *has_as_opposing_force*
Prompt: "Name the opposing forces in "<*current-instance*>
Control-type: *multiple-names*
Other ontology actions: <*property-value*> instance-of Opposing_force
. . .

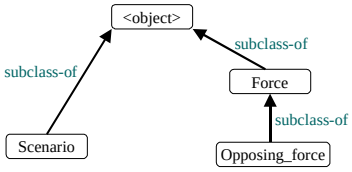


Figure 6: Fragment of the elicitation script for "Scenario"

7 Agent Teaching and Rule Learning

After an object ontology has been developed, the subject matter expert starts teaching the agent how to solve problems through successive reductions and compositions. This process is explained in [Tecuci *et al.*, 2000b]. Here we only briefly review it in order to have a complete description of the Disciple methodology. In essence, the subject matter expert starts from the domain models that

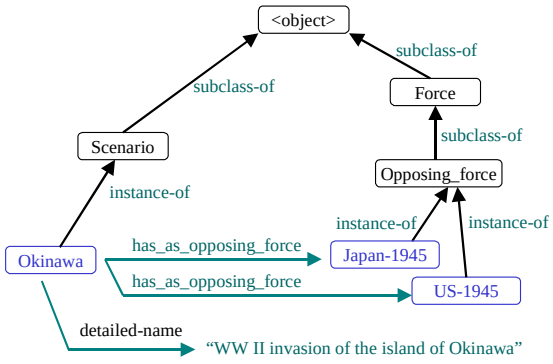


Figure 7: The result of the elicitation script from figure 6

have been previously prepared in collaboration with the knowledge engineer, as presented in section 3. The left hand side of Figure 1 shows an example of the task-reduction modeling of the problem solving process. Each abstract task reduction step (consisting of a task, a question, an answer, and a subtask) is expanded into a training example for the Disciple agent, as illustrated in Figure 8. Each task is now represented by a name phrase and a set of feature-value pairs. The answers are also made more specific. From each such task reduction example the agent learns a general task reduction rule that will allow it to apply a similar task reduction operation in future problem-solving situations. For instance, the rule learned from the second task reduction step in figure 8 is represented in figure 9. The process of learning such a general task reduction rule is a mixed-initiative one. First the subject matter expert and the agent collaborate in finding a formal justification of why the current task reduction is correct. Then, based on the found justification, the agent generalizes the example into a task reduction rule. As shown in figure 9, the learned rule is an IF-THEN structure with two applicability conditions, a plausible lower bound condition and a plausible upper bound condition. These two conditions represent a plausible version space for the exact applicability condition of the rule. Through further learning, the two conditions converge toward one another and toward this exact condition. An important thing to notice is that the rule's conditions are expressed in term of the concepts and the features from the object ontology. In general, the rule's conditions could be much more complex expressions than the ones illustrated in figure 9.

8 Ontology Learning

Ontology learning is becoming an important research issue [Staab *et al.* 2000]. It also plays an important role in the Disciple agent development methodology. During the process of defining or explaining specific task reductions or compositions, when training the agent, the subject matter expert may need to refer to objects or object features that are not yet part of the ontology. From these specific instances Disciple-RKF will learn general ontological elements. For example, the subject matter expert may point to a specific feature of an object, as being responsible for the failure of a certain task reduction step. In such a case the agent will learn a general object feature definition from that specific feature. Any object feature definition specifies a domain (a concept that represents the set of objects that could have that feature) and a range (another concept that represents the set of possible values of that feature).

Disciple-RKF generates a plausible version space for the domain concept, and another one for the range concept. These version spaces are similar to the plausible version space condition of the rule shown in figure 9. After the versions spaces are generated, Disciple initiates a feature refinement experimentation session with the goal of reducing the plausible version spaces of the feature's domain and range.

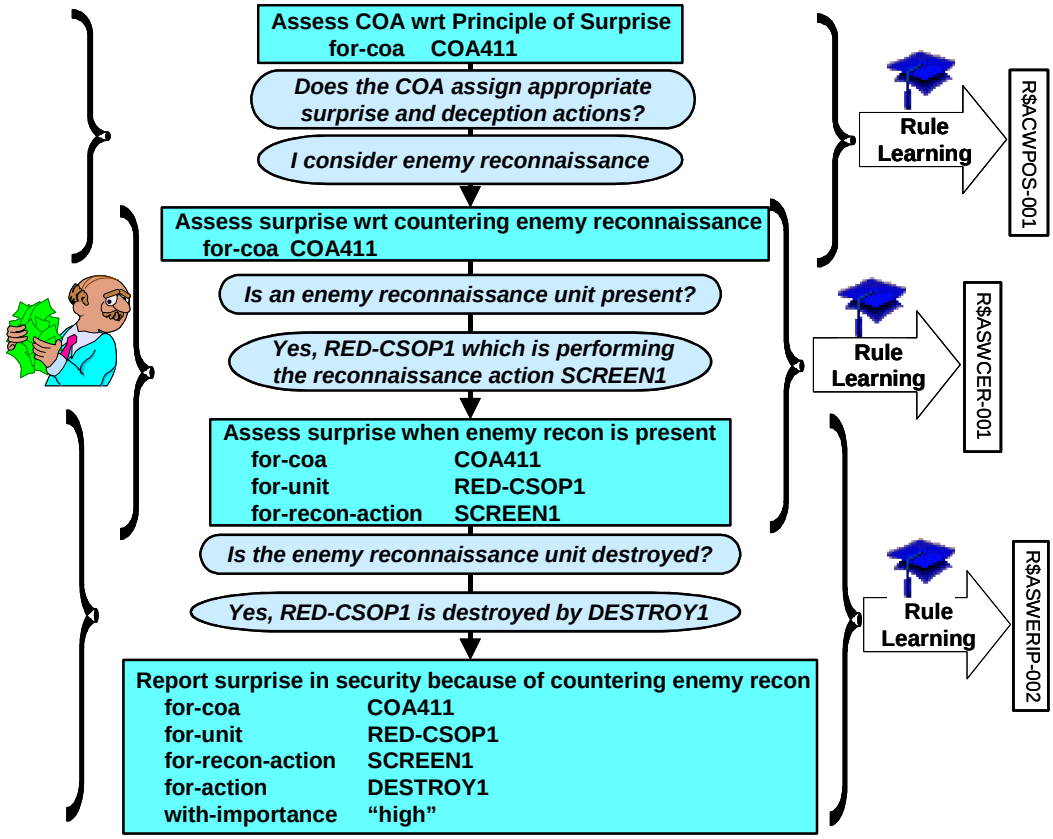


Figure 8: Sample teaching and learning scenario

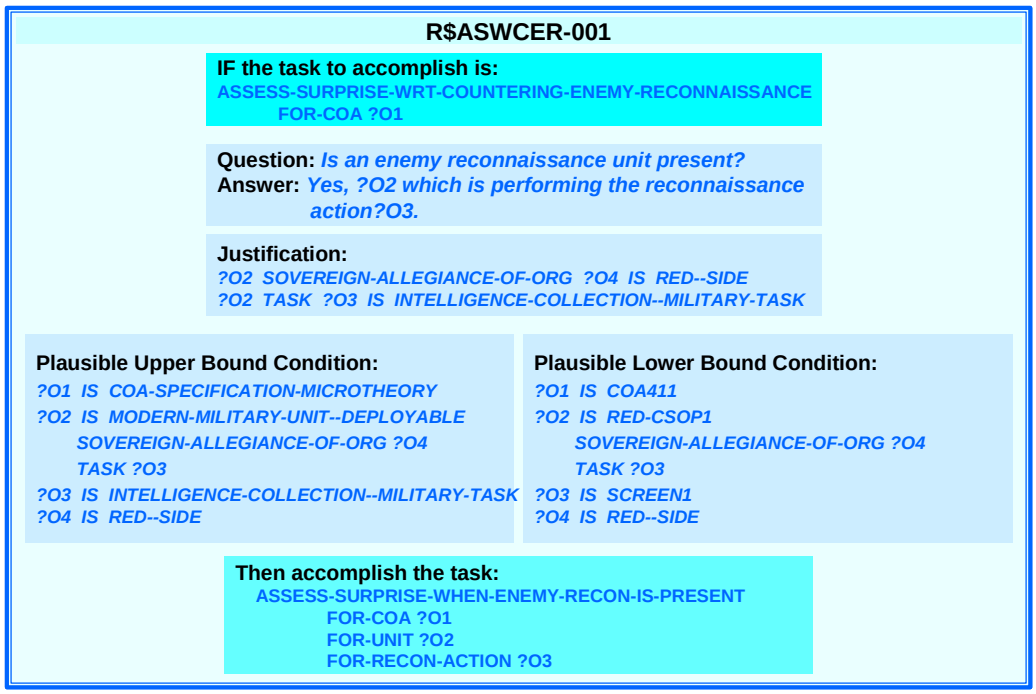


Figure 9: Sample task reduction rule learned by Disciple

9 Knowledge Base Export

Figure 10 illustrates the synergistic relationship between the Disciple-RKF agent development tool and an external knowledge server, such as CYC. To develop a knowledge-based agent with Disciple-RKF one starts by importing an initial object ontology from the CYC knowledge server, as discussed in section 4. Then the subject matter expert interacts with Disciple, teaching it to solve problems, and thus developing the knowledge base of Disciple to incorporate the expertise of the subject matter expert. After the Disciple knowledge base has been developed, it is exported back into CYC, as a separate CYC microtheory. This is an automatic translation process that does not raise any problems because CYC's knowledge representation is more powerful than that of Disciple-RKF. Then this CYC microtheory can be semantically integrated with the rest of the CYC knowledge repository, by the developers of CYC. This semantic integration is a difficult task, but it is facilitated in this case by the fact that the initial Disciple ontology has been imported from CYC, to begin with.

We have performed a preliminary experiment during which the knowledge base of Disciple corresponding to the Course of Action challenge problem has been automatically translated into a CYC microtheory. Then, using its inference engine, CYC generated the same critiques of a course of action as Disciple. The integration model described above can be adapted for any other knowledge server, with only minor modifications. For instance, in the case of an OKBC knowledge server, only the object ontology of Disciple will be exported because the rules cannot be represented using the OKBC frame-based knowledge model.

10 Experimental Results

Successive versions of the Disciple approach and other competing knowledge base development approaches have been evaluated in several intensive studies requiring the rapid development and maintenance of knowledge bases for solving the workaround challenge problem (consisting of planning the repair of damaged bridges and roads [Jones, 1998]), and the COA challenge problem (consisting of generating critiques of military courses of action [Jones, 1999]). These evaluations were performed by Alphatech, as part of the DARPA's HPKB program, and involved, in addition to Disciple, the following teams and approaches: 1) Teknowledge and Cycorp that used the CYC system [Lenat, 1995].

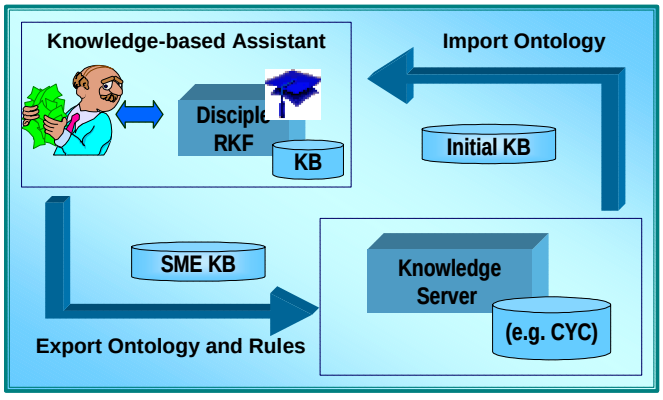


Figure 10: The synergy between Disciple-RKF and the knowledge servers

They achieved rapid knowledge base development through extensive re-use of CYC's carefully developed ontology, wide-ranging common-sense knowledge, and general inferential capabilities. 2) The EXPECT group from USC-ISI. This group developed knowledge bases with wide problem coverage and expert-level performance, using the knowledge acquisition tools of EXPECT that assist a knowledge engineer in debugging and refining a knowledge base [Kim and Gil, 1999]. 3) The Loom/PowerLoom group from USC-ISI that used novel case-based reasoning techniques for the COA challenge problem, in conjunction with the PowerLoom [MacGregor, 1999] representation system and an imported ontology. 4) The AIAI group from the University of Edinburgh that developed a high performance knowledge base for the workaround challenge problem by designing a planning ontology in CYC.

In these experiments all the approaches demonstrated very good results and relative technology strengths. However, the Disciple approach has achieved the highest rates of knowledge acquisition and the best problem solving performance, while the generated solutions and justifications where judged as being very intelligible.

The first evaluation concerned the workaround challenge problem and lasted for 17 days. At the beginning of the evaluation Disciple had an incomplete knowledge base consisting of 723 object concepts, 100 tasks, and 121 task reduction rules. Out of the 723 concepts 126 were imported from LOOM (an OKBC compliant knowledge server). They included elements of the military unit ontology, as well as various characteristics of military equipment (such as their tracked and wheeled military load classes). The extent of knowledge import was more limited than it could have been because the LOOM's ontology was developed at the same time as that of Disciple, and we had to define concepts that have later been also defined in LOOM and could have been imported. In any case, importing those concepts proved to be very helpful, and has demonstrated the ability to reuse previously developed knowledge.During the 17 days of the evaluation, the knowledge base of Disciple was increased with 147 object concepts, 104 tasks, and 87 complex task reduction rules. The performance of the developed knowledge-based agent was judged by the evaluators as being at the level of a human expert.

The second evaluation concerned the COA challenge problem and lasted 8 days. In this case the initial ontology was imported from CYC. During the evaluation period the knowledge base of Disciple was increased by 46%, which represents an even higher daily rate of knowledge acquisition than in the first experiment. Also, in addition to generating most of the critiques expected by the evaluators, Disciple generated many new critiques. The final knowledge base contained 801 concepts, 444 object and task features, 360 tasks and 342 rules. Also, each input problem (the description of a course of action) was represented with around 1500 facts. Currently Disciple is further developed and evaluated at the US Army War College, being used by subject matter experts to develop knowledge bases for the identification of strategic center of gravity candidates.

11 Conclusions and Future Research

We have presented an approach to rapid development of knowledge-based agents by subject matter experts that is based on ontology reuse and development.

In addition to the further development of the methods presented in this paper, future research on ontologies and information sharing will consist in extending the Disciple approach (Tecuci, 1998) and the supporting tools to allow several experts to collaborate in building different parts of a larger knowledge base. In particular, we plan to develop a distributed architecture for collaborative knowledge base development, as shown in Figure 11.

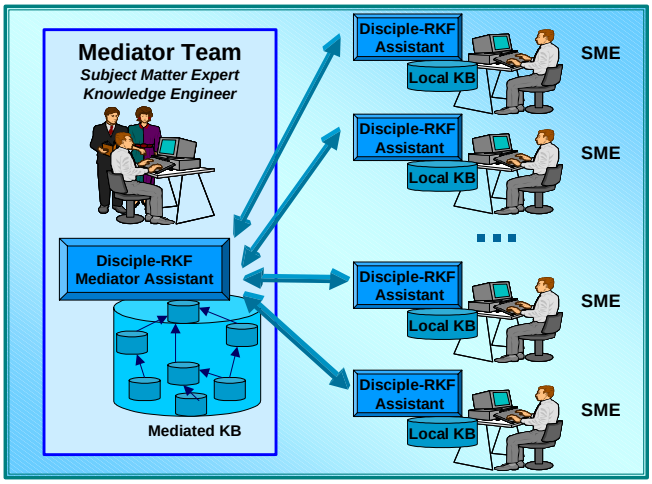


Figure 11: Collaborative knowledge base development

The right hand side of Figure 11 represents a team of subject matter experts that collaborate to rapidly build an integrated knowledge base. Each individual subject matter expert works with a personal Disciple-RKF agent to build a part of the integrated knowledge base. These separately developed knowledge bases are periodically integrated into a single knowledge base by the mediator team that includes a knowledge engineer, a subject matter expert, and a Disciple agent specialized in knowledge integration. The mediator team not only integrates the knowledge bases, but also mediates the collaboration between all the subject matter experts.

Several features of the proposed approach facilitate collaborative knowledge base development. First, the knowledge base is structured into an object ontology that defines the terms of the representation language, and set of task reduction rules that are expressed using these terms. As a consequence, the subject matter experts have to agree on the shared object ontology but they can develop the rules independently.

Second, the knowledge base to be built is divided into parts that are as independent as possible, with each subject matter expert responsible for the development of a different part. The mediator team coordinates the partitioning and the integration of the knowledge base, and facilitates a consensus among the subject matter experts concerning the developed knowledge that is to be shared.

Third, the integrated knowledge base consists of a hierarchy of component knowledge bases that are each internally consistent, but may contain portions that supersede or contradict portions from other knowledge bases. This corresponds to the fact that the knowledge model of a subject matter expert is internally consistent but it may contain knowledge that contradicts aspects of the knowledge model of another subject matter expert. This knowledge base organization not only facilitates knowledge acquisition from multiple subject matter experts, but also leads to a knowledge base that can provide solutions to problems from different points of view.

Acknowledgements

This research has been performed in the GMU Learning Agents Laboratory and was sponsored by the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0546, by the Air Force Office of Scientific Research (AFOSR) under grant no. F49620-00-1-0072, and by the US Army. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Dorin Marcu, Michael Bowman, Cristina Cascaval, and other members of the LALAB have contributed to successive versions of Disciple. In addition to Michael Bowman, Tony Lopez and Jim Donlon, from the Center for Strategic Leadership of the US Army War College, have contributed to the application of Disciple to the center of gravity challenge problem. The anonymous reviewers of this paper provided insightful comments that helped us to improve it.

References

[Boicu *et al.*, 2000] Mihai Boicu, Gheorghe Tecuci, Dorin Marcu, Michael Bowman, Ping Shyr, Florin Ciucu, and Cristian Levcovici. Disciple-COA: From Agent Programming to Agent Teaching. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA, Morgan Kaufmann, 2000.

[Buchanan and Wilkins, 1993] Bruce G. Buchanan and David C. Wilkins (editors). *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems.* Morgan Kaufmann, San Mateo, CA., 1993

[Burke, 1999] Murray Burke. *Rapid Knowledge Formation (RKF) Program Description*, http://dtsn.darpa.mil/iso/index2.asp?mode=9

[Chaudhri *et al.* 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Daniel P. Park, and James P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Menlo Park, CA: AAAI Press, pages 600 – 607, 1998.

[Cohen *et al.*, 1998] Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning and Murray Burke. The DARPA High-Performance Knowledge Bases Project, *AI Magazine*, 19(4), 25-49, 1998.

[Corcho and Gomez-Perez, 200] Oscar Corcho and Asuncion Gomez-Perez. Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In *Proceedings of the ECAI 2000 Workshop on Application of Ontologies and Problem-Solving Methods, Berlin, 2000*.

[Farquhar *et al.*, 1996] Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada, 1996.

[Fridman *et al.*, 2000] Natalya Fridman Noy, Ray W. Fergerson, Mark A. Musen. The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In *Proceedings of the European Knowledge Acquisition Workshop*, pages 17-32, 2000.

[Gilles *et al.*, 1996] MAJ Phillip Kevin Giles, CPT Thomas P. Galvin. *Center of Gravity: Determination, Analysis, and Application*, U.S. Army War College, Carlisle Barracks, PA, 1996.

[Jones, 1998]. Eric Jones. *HPKB Year1 End-to-End Battlespace Challenge Problem Specification*. Burlington, 1998.

[Jones, 1999]. Eric Jones. *HPKB Course of Action Challenge Problem Specification*. Alphatech, Burlington, 1998.

[Kim and Gil, 1999]. Jihie Kim and Yolanda Gil. Deriving Expectations to Guide Knowledge Base Creation. In *Proc. of the Sixteenth National Conference on Artificial Intelligence*,235-241, Menlo Park, CA: AAAI Press, 1999.

[Lenat, 1995] Douglas B. Lenat. CYC: A Large-scale investment in knowledge infrastructure. In *Communications of the ACM* 38(11): 33-38, 1995.

[MacGregor, 1999] Robert MacGregor. *Retrospective on LOOM.* Available online: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html, 1999.

[McGuinness *et al.*, 2000] Deborah L. McGuinness, Richard Fikes, James Rice, Steve Wilder. The Chimaera Ontology Environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Menlo Park, CA, AAAI Press, pages 1123-1124, 2000.

[Staab *et al*. 2000] Steffen Staab, Alexander Maedche, Claire Nedellec, and Peter Wiemer-Hastings (eds.) *Proceedings of the First Workshop on Ontology Learning OL'2000*, Berlin, Germany, August 25, 2000.

[Tecuci, 1998] Gheorghe Tecuci. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies.* London, England: Academic Press, 1998.

[Tecuci et al. 2000a] Gheorghe Tecuci, Mihai Boicu, Kathryn Wright, Seok-Won Lee, Dorin Marcu, and Michael Bowman. A Tutoring Based Approach to the Development of Intelligent Agents. In Teodorescu, H.N., Mlynek, D., Kandel, A. and Zimmermann, H.J. (editors). *Intelligent Systems and Interfaces*, Kluwer Academic Press. 2000.

[Tecuci et al. 2000b] Gheorghe Tecuci, Mihai Boicu, Michael Bowman, Dorin Marcu, Ping Shyr, and Cristina Cascaval. 2000 "An Experiment in Agent Teaching by Subject Matter Experts," *International Journal of Human-Computer Studies* 53: 583-610.

Towards a principled approach to semantic interoperability

Jérôme Euzenat
INRIA Rhône-Alpes
655 avenue de l'Europe, 38330 Montbonnot Saint-Martin (France)
Jerome.Euzenat@inrialpes.fr

Abstract

Semantic interoperability is the faculty of interpreting knowledge imported from other languages at the semantic level, i.e. to ascribe to each imported piece of knowledge the correct interpretation or set of models. It is a very important requirement for delivering a worldwide semantic web. This paper presents preliminary investigations towards developing a unified view of the problem. It proposes a definition of semantic interoperability based on model theory and shows how it applies to already existing works in the domain. Then, new applications of this definition to family of languages, ontology patterns and explicit description of semantics are presented.
Keywords: Semantic interoperability, ontology sharing, knowledge transformation, ontology patterns.

1 Introduction

The vision of a "semantic web" [Berners-Lee, 1998; Berners-Lee *et al.*, 2001] is realized by the annotation of web pages, containing informal knowledge as we know it now, with formal knowledge. These annotations can reference each other and depend on ontologies and background knowledge. Taking advantage of the semantic web requires to be able to gather, compare, transform and compose the annotations. For several reasons (legacy knowledge, ease of use, heterogeneity of devices and adaptability, timelessness), it is not likely that this formal knowledge will be encoded in the very same language. The interoperability of formal knowledge languages must then be studied in order to interpret the knowledge acquired through the semantic web.

The problem of comparing theory is well known but it takes a fantastic importance in the context of the semantic web.

Semantic interoperability is the faculty of interpreting the annotations at the semantic level, i.e. to ascribe to each imported piece of knowledge the correct interpretation or set of models. It will be further characterized below by considering that the final transformed knowledge must have the transformation of the consequences of the initial knowledge as consequences.

There are several approaches to semantic interoperability [Masolo, 2000; Ciocoiu and Nau, 2000; Stuckenschmidt and Visser, 2000]. Although, they are not stated in the same terms, we believe that there can be a unified view of comparison and transformation at a semantic level that can be applied to these approaches.

We first provide some definitions of the concepts at work here (language, representation, semantics and transformation) and a classification of possible interoperability requirements. Then the already available approaches to semantic interoperability are considered and rephrased in the context of model-theory. Afterwards, we turn to consider three possible approaches for the semantic web (and especially when the languages are different): family of languages, ontology patterns and explicit semantics representation. We show how the contribution of these techniques to semantic interoperability can be expressed in comparable terms.

2 Principles

2.1 Language, semantics, transformation

For the simple purpose of the present paper, a language L will be a set of expressions. A representation (r) is a set of expressions in L. No distinction will be made between ontologies, background knowledge and formal annotations: they will all be representations.

There have been many studies of knowledge representation language semantics [Nebel, 1990]. The semantics is generally defined in model theory by using simple set theory. Usually, an interpretation function I, to a domain of interpretation D, is defined iteratively over the structure of the language L. The interpretation function is compositional, i.e. it builds the meaning of an expression from that of its sub-expressions (or components). The expressions δ in a language L are said to be satisfied by interpretation I if they meet a certain condition (usually that $I(\delta)$ belongs to a distinguished subset of the domain). In this framework, a model of a set of assertions $r \subseteq L$, is an interpretation I satisfying all the assertions in r. An expression δ is said to be a consequence of a set of expression r if it is satisfied by all models of r (this is noted $r \models_L \delta$).

A computer has to find if a particular expression (e.g. a

query) is the consequence of a set of expression (e.g. a knowledge base). To that extent, executable systems (called provers) are developed which can be grounded on inference rules or more classical programs. From a set of axioms $r \subseteq L$, they establish if the expression $\delta \in L$ is a theorem (noted $r \vdash_L \delta$). These provers are said correct if any theorem is a consequence of the axioms and complete if any consequence of the axioms is a theorem. However, depending on the language and its semantics, the decidability (i.e. the existence of such provers) is not ensured and, even in this event, the algorithmic complexity of such provers can be prohibitive.

Hence, system developers must establish a trade-off between expressivity and complexity of representation languages or completeness of the prover. This choice has led to the definition of languages with a low expressivity (like simple conceptual graphs or object-based representations) or modular family of representation languages (like description logics). As a consequence, there are many different representations languages that can be used in the context of the semantic web. Therefore, if some annotation, ontology or background knowledge is found on the semantic web, it might have to be translated from a language to another.

Transformations are applied to representations in order to import them from one language to another. The transformations are functions $\tau : L \longrightarrow L'$ (L' can be L). These transformations must be computable (and so they are syntactic mechanisms) and in the context of the semantic web, they can be implemented as XSLT (or a similar language) stylesheets [James Clark (ed.), 1999]. These transformations can be composed into more complex transformations.

We will consider here the problem of ensuring the interoperability of representations through transformations. There are several levels at which interoperability can be accounted for.

2.2 Levels of interoperability

When trying to assess the understanding of an expression coming from a system by another one, there are several possible levels of interoperability:

- encoding: being able to segment the representation in characters;
- lexical: being able to segment the representation in words (or symbols);
- syntactic: being able to structure the representation in structured sentences (or formulas or assertions);
- semantic: being able to construct the propositional meaning of the representation;
- semiotic: being able to construct the pragmatic meaning of the representation (or its meaning in context).

This layered presentation is arguable in general; it is not as strict as it seems. It makes sense because each level cannot be achieved if the previous ones have not been completed. In the context of the semantic web, it can be assumed that the three first levels can be easily achieved by the use of XML or RDF.

The properties of transformations can be set at these various levels. There are many kinds of properties (e.g. at the syntactic level we could care of just preserving the elements, or their ordering or both). Here are some of their expressions:

- lexical: given a mapping σ between terms of a particular language (that can be the terms in a structured formal language or the lexical units of a natural language). For instance, one can ask that this mapping preserves *synsets* (connected components of the synonymy graph, S), i.e. $\forall t, t' \in L, S(t) = S(t') \Rightarrow S(\sigma(t)) = S(\sigma(t'))$.
- syntactic: Order-preservation, for instance, will require that, given two order relations $\leq_L$ and $\leq_{L'}$, if $r \leq_L s$, then $\tau(r) \leq_{L'} \tau(s)$.
- semantic: Consequence preservation requires that

  (1) $$\forall \delta, r \models_L \delta \Rightarrow \tau(r) \models_{L'} \tau(\delta);$$

  It is noteworthy that consequence preservation is not trivially granted by syntactic preservation, e.g. addition in the structure. As a matter of fact, adding a class or an attribute in a frame-based knowledge base is structure preserving (it preserve both elements and their order) though the second addition is not consequence preserving.
- semiotic: interpretation preservation (let Σ be the interpretation rules and $\models^i$ the interpretation relation for person i, $\forall \delta, \forall i, j, r, \Sigma \models^i \delta \Rightarrow \tau(r), \tau(\Sigma) \models^j \tau(\delta)$)). This consists in mapping signs to equivalent signs with respect to the expected interpretation of a reader. These aspects can be related to rhetoric [Rutledge *et al.*, 2000] or pragmatics (i.e. properties not directly relevant to a compositional view of semantics but which interfere with sheer semantic interpretation).

We do not pretend that these properties must be satisfied in the semantic web. There can be situations where only some moderate preservation of meaning or content is sufficient. However, characterizing the exact properties of the available transformations will be very useful.

The present paper will only consider semantic interoperability and especially how the formula 1 can be satisfied in various contexts. This scheme, in which a first statement in a system constrains another one in another system, is necessary for relating statements across languages while pure logical statements can be used (e.g. in modal logics of knowledge [Fagin *et al.*, 1995]) when the language is shared across agents.

3 Related work

In the context of first-order logic ($\mathcal{FOL}$), Claudio Masolo [Masolo, 2000] has investigated the relations between logical theories. He first considers the deductive closure of sets of axioms ($Cn(r) = \{\delta | r \vdash_{\mathcal{FOL}} \delta\}$) and the relationships between these representations derived from the five possible containment relations on their deductive closure. Since this can only be applied to theories with coinciding vocabularies, he goes on by considering the definitional extension (noted $r'|^r$) of a representation r', by the definition of the terms (predicate and function symbols) of a representation r,

in functions of the terms in r'. In such a case, importing a representation from an ontology into another is simply $r'|^r \cup r$. And this warrants that:

$$\forall \delta \in \mathcal{FOL}, r \models_{\mathcal{FOL}} \delta \Rightarrow r'|^r \cup r \models_{\mathcal{FOL}} \delta$$

So, consequence is trivially preserved (the importance of this notion in [Masolo, 2000] is related to inter-expressibility of theories). Of course, there can be no definitional extension of r' by r.

Claudio Masolo also considers translations (ϕ) which are transformations preserving the structure of formulas (i.e. atomic formulas are replaced by arbitrary formulas and the rest of the transformation is defined by induction on the structure of formulas) and renaming (σ) which are translations only affecting the name of predicates and functions. These translations can be though of as our transformation τ. The theories can be compared based on mutual translations:

$$r \approx_{\tau,\tau'} r' \text{ iff } r' \vdash_{\mathcal{FOL}} \tau(r) \text{ and } r \vdash_{\mathcal{FOL}} \tau'(r')$$
$$r \prec_{\tau,\tau'} r' \text{ iff } r' \vdash_{\mathcal{FOL}} \tau(r) \text{ and } r \nvdash_{\mathcal{FOL}} \tau'(r')$$
$$r \succ_{\tau,\tau'} r' \text{ iff } r' \nvdash_{\mathcal{FOL}} \tau(r) \text{ and } r \vdash_{\mathcal{FOL}} \tau'(r')$$
$$r \times_{\tau,\tau'} r' \text{ iff } r' \nvdash_{\mathcal{FOL}} \tau(r) \text{ and } r \nvdash_{\mathcal{FOL}} \tau'(r')$$

For our purposes, this is equivalent to define:

$$r \preceq_{\phi} r' \text{ iff } r' \vdash_{\mathcal{FOL}} \phi(r)$$

and thus (by induction on the formula structures):

$$\forall \delta, r \models_{\mathcal{FOL}} \delta \Rightarrow \phi(r) \models_{\mathcal{FOL}} \phi(\delta)$$

Claudio Masolo shows that the equivalence relations through translations and the equivalence through definitional extensions are indeed equivalent (in terms of deductive closures). These relations have their equivalent characterization in model theory:

$$r \approx_{\tau,\tau'} r' \text{ iff } \forall \delta, r \models_{\mathcal{FOL}} \delta \Leftrightarrow r' \models_{\mathcal{FOL}} \delta$$
$$r \prec_{\tau,\tau'} r' \text{ iff } \forall \delta, r \models_{\mathcal{FOL}} \delta \Rightarrow r' \models_{\mathcal{FOL}} \delta$$
$$r \succ_{\tau,\tau'} r' \text{ iff } \forall \delta, r' \models_{\mathcal{FOL}} \delta \Rightarrow r \models_{\mathcal{FOL}} \delta$$
$$r \times_{\tau,\tau'} r' \text{ iff } \exists \delta \in L, \delta' \in L', r \models_{\mathcal{FOL}} \delta, r' \models_{\mathcal{FOL}} \delta', r \not\models_{\mathcal{FOL}} \delta' \text{ and } r' \not\models_{\mathcal{FOL}} \delta$$

He demonstrates that, provided with completeness of first order logic, the straightforward semantics characterization for the theory with coinciding vocabularies (and theories equivalent through renaming alone) is equivalent to the syntactic one. The formulation of the equivalent of definitional extensions is related to the notion of "coalescent models" which is not detailed here.

A last contribution of [Masolo, 2000] is the comparison of logics whose set of models coincide while they do not use translatable primitives (e.g. the geometry based on points and those based on spheres). The notion of model-structure transformations (i.e. transformation applying at a semantic level) are introduced.

Ciocoiu [Ciocoiu and Nau, 2000], takes into account the implicit knowledge (K) that is not formally expressed in the ontologies but should be taken into account by building the models. Their framework uses a first-order logical language as a pivot languages in which both languages can be translated (and from which models can be extracted) and an ontology of explicit assumptions expressed in the formal language. The goals of this work is the translation-checking (i.e. knowing if a representation is the translation of another, i.e. if it has the same set of models). This is theoretically achieved by comparing the corresponding the sets of extracted models.

This can be further refined by considering two background knowledge sets (K and K'), the knowledge transformations can be justified in our framework without the common ontology and logical translation:

$$\forall \delta, K, r \models_{\mathcal{FOL}} \delta \Rightarrow K', \tau(r) \models_{\mathcal{FOL}} \tau(\delta)$$

In a similar vein, [Stuckenschmidt and Visser, 2000] introduced the idea that, beside the correct syntactic transformation required for semantic interoperability, there is room for several completeness levels that must be taken into account. The most basic level is sheer translation of what is (syntactically) transcriptible from the source representation. A second level consists of ensuring that whatever is a consequence of the source representation that can be expressed in the target language is indeed translated. In case of a more expressive source language this might require the use of a prover in order to deduce these formulas that can be represented by the target (but more generally, a prover might be required whatever the expressivity of either languages). This means that, given a sheer syntactic transformation τ, one must build a semantic transformation $\overline{\overline{\tau}}$ such that:

$$\forall \delta \in L', \tau(Cn(r)) \models_{L'} \delta \Rightarrow \overline{\overline{\tau}}(r) \models_{L'} \delta$$

or

$$\forall \delta \in L, r \models_{L} \delta \Rightarrow \overline{\overline{\tau}}(r) \models_{L'} \tau(\delta)$$

The translations of [Masolo, 2000] are such semantic transformations.

A further refinement, well represented in [Ciocoiu and Nau, 2000], is the explicitation of implicit knowledge, that can be added as background for the translated theory. In the context of geographical information integration [Visser *et al.*, 2000], the authors have integrated the domain ontologies by providing translations from the source ontologies in a target language and by reclassifying the corresponding concepts (grounded on their descriptions) with regard to each other.

OntoMorph [Chalupsky, 2000] is a system of syntactic transformation of ontologies with a syntactic transformation language not very different from XSLT. It however is integrated with a knowledge representation system (PowerLoom) which provides the opportunity to have semantically-grounded rules in the transformations. The system can query assertions for not only being syntactically in the source representation, but also for being a consequence of this initial representation (as soon as PowerLoom is semantically complete). This is a generic implementation of what is proposed in [Stuckenschmidt and Visser, 2000]. Of course, this option requires to use PowerLoom as an initial pivot language and the problem of translation arises when transforming from the source representation to the PowerLoom representation.

Semantic, knowledge or ontology patterns [Staab *et al.*, 2000; Clark *et al.*, 2000; Stuckenschmidt, 2000] have recently been introduced as the equivalent, in the ontology engineering field, of the design patterns (or rather frameworks) in software engineering. They are used for factoring out notions that are common across, and despite, languages. Instead of considering a knowledge representation construct in isolation, it considers a set of constructs and their interrelations satisfying a particular function (e.g. how to deal with part-whole relations, how to deal with class specialization). To that extent, ontology patterns offer a language $\mathcal{P}$ for expressing the required constructs and the constraints that hold between them. A pattern p is usually made of a set of terms T, a set of grammar rules for articulating them G and a set of constraints C. Implementations consist in instantiating the patterns, i.e. mapping the constructions and constraints to the concrete language. The mapping μ is specified in terms of signature morphism between the pattern p and an actual language L such that:

$$\forall \delta, p \models_{\mathcal{P}} \delta \Rightarrow \mu(p) \models_{L} \mu(\delta)$$

These contributions have especially considered semantic interoperability within the same language (using different sets of axioms or ontologies) or generic to specific languages (through pattern mapping). We will now take a look at several proposals for expressing semantic interoperability across different languages as it shall happen on the semantic web.

4 Language family approach

In the words of Tim Berners-Lee, the semantic web requires a set of languages of increasing expressiveness and anyone can pick up the right language for each particular semantic web application.

A modular family of languages is a set $\mathcal{L}$ of languages that have a similar kind of formulas (e.g. build from a subset of the same set of formula constructors) and the same kind of semantic characterization (i.e. if a formula belongs to two languages, it is interpreted in the same way in both). It is then easy to transform a representation from one language to another and one can take advantage of more efficient provers or more expressive languages.

This is what have been developed by the description logic community over the years: a family of representation languages with known decidability, complexity and equivalence results [Donini *et al.*, 1994]. It has been experimented for the web with the "Description Logic Mark-up Language"[1] (DLML) that we have developed.

DLML is not a language but rather a modular system of document type descriptions (DTD) encoding the syntax of many description logics. It takes advantage of the modular design of description logics by describing individual constructors separately. The specification of a particular logic is achieved by declaring the set of possible constructors and the logic's DTD is automatically build up by just assembling those of elementary constructors. The actual system contains the description of more than 40 constructors and 25 logics.

[1] `http://co4.inrialpes.fr/xml/dlml/`

To the DLML language is a associated a set of transformations (written in XSLT) allowing to convert a representation from a logic to another. The simplest transformation is the transformation from a logic to another syntactically more expressive one (i.e. which adds new formulas). The transformation is then trivial, but yet useful, because the initial representation is valid in the new language, it is thus identity:

$$\forall \delta \in \mathcal{AL}, r \models_{\mathcal{AL}} \delta \Rightarrow r \models_{\mathcal{ALC}} \delta$$

This trivial interpretation of semantic interoperability is one strength of the "family of languages" approach because, in the present situation, nothing has to be done for gathering knowledge. For this case, one can define the relation between two languages L and L' as $L \overline{\ll} L'$ which has to comply with $L \subseteq L'$. We can then define $L \equiv L'$ as equivalent to $L \overline{\ll} L'$ and $L' \overline{\ll} L$.

We can further define $L \overline{\triangledown} L'$ by $L \overline{\ll} L \overline{\triangledown} L'$ and $L' \overline{\ll} L \overline{\triangledown} L'$ and there exists no other language L'' such that $L \overline{\ll} L'' \overline{\ll} L \overline{\triangledown} L'$ and $L' \overline{\ll} L'' \overline{\ll} L \overline{\triangledown} L'$. For all L and L', $L \overline{\triangledown} L'$ and $L \overline{\wedge} L'$ have to satisfy $L \cup L' \subseteq L \overline{\triangledown} L'$ and $\overline{\wedge} L' \subseteq L \cap L'$ (in the case of term-based languages such as description logics we have $L \overline{\wedge} L' = L \cap L'$, but not necessarily $L \overline{\triangledown} L' = L \cup L'$, see figure 1). This defines the syntactic structure of $\mathcal{L}$.

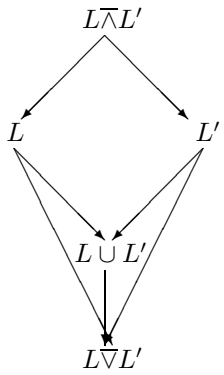


Figure 1: The relations between syntactic languages

If $L \overline{\not\ll} L'$, the transformation is more difficult. The initial representation r can be restricted to what is (syntactically) expressible in L': $\overline{r}$. However, this operation (which is correct) is incomplete because it can happen that a consequence of a representation expressible in L' is not a consequence of the expression of that representation in L':

$$\exists \delta \in L'; \overline{r} \not\models_{L'} \delta \text{ and } r \models_{L} \delta$$

To solve this problem, as stated in [Visser *et al.*, 2000], it is necessary to deduce from r in L whatever is expressible in L'. Let $\overline{\overline{r}} = \overline{Cn(r)}$ be this expression. It is such that

$$\forall r \subseteq L, \forall \delta \in L \overline{\wedge} L', r \models_{L} \delta \Rightarrow \overline{\overline{r}} \models_{L'} \delta$$

The preceding proposal is restricted in the sense that it only allows, in the target language, expressions expressible in the source language, while there are equivalent non-syntactically comparable languages. This is the case of the description

logic languages $\mathcal{ALC}$ and $\mathcal{ALUE}$ which are known to be equivalent while none has all the constructors of the other. For that purpose, one can define $L\overline{\overline{\ll}}L'$ if and only if the models are preserved, i.e. $\exists\overline{\overline{\tau}}$;

$$\forall r \subseteq L, \forall\langle I, D\rangle; \langle I, D\rangle \models_{L'} \tau(r), \Rightarrow \langle I, D\rangle \models_{L} r$$

Similarly, $L\overline{\overline{\equiv}}L'$ if and only if $L\overline{\overline{\ll}}L'$ and $L'\overline{\overline{\ll}}L$. Moreover, $L\overline{\overline{\nabla}}L'$ is defined by a language such that $L\overline{\overline{\ll}}L\overline{\overline{\nabla}}L'$ and $L'\overline{\overline{\ll}}L\overline{\overline{\nabla}}L'$ and there exists no other language L'' such that $L\overline{\overline{\ll}}L''\overline{\overline{\ll}}L\overline{\overline{\nabla}}L'$ and $L'\overline{\overline{\ll}}L\overline{\overline{\nabla}}L'$. [2]

The $\overline{\overline{\tau}}$ transformation is not easy to produce (and it can generally be computationally expensive) but we show, in §6 how this could be practically achieved.

Another possibility is to define $\widetilde{\ll}$ as the existence of an isomorphism between the models of r and those of $\tau(r)$

$$\exists\tau; \forall\langle I, D\rangle, \exists\langle I', D'\rangle; I, r \models_{L} \delta \Rightarrow I', \tau(r) \models_{L'} \tau(\delta)$$

This also ensures that $r \models_{L} \delta \Rightarrow \tau(r) \models_{L'} \tau(\delta)$.

This provides to the family of languages a structure based on semantics.

5 Pattern-based approach

The generic pattern based approach provides patterns of constructs involved in a language. In the present article, a pattern $p \in \mathcal{P}$ is characterized by a set of constructions that can be mapped to that of a language and an interpretation of these constructions that must be preserved by the mapping to a concrete language (this is also seen as a constraint). For instance, a CONJ(.) pattern is interpreted over sets as the intersection of conjunct constructions. It is mapped to the `AND` and `ANDROLE` operators in description logics or the class constructor in frame-based languages. The patterns do not provide a direct way to ensure the interoperability between two languages.

However, translating between two languages which share some patterns should be easier than the general case. As a matter of fact, if, as is assumed, the mappings preserve meaning (i.e. $p \models_{\mathcal{P}} \delta \Rightarrow \mu(p) \models_{L} \mu(\delta)$). Then, in the case of reversible or bijective mappings (i.e. such that

$$\exists\mu^{-1}; r \models_{L} \delta \Leftrightarrow p = \mu^{-1}(r) \models_{\mathcal{P}} \mu^{-1}(\delta))$$

it is possible to ensure that the transformation from L to L' made by $\mu' o \mu^{-1}$ indeed preserves meaning. More precisely, since the constraints are implemented and satisfied by both systems, only the mapping of constructors and grammar are required to be bijective.

Of course, not all such mappings are bijective, though there should be numerous cases in which the mapping is indeed bijective: it should be possible to establish when μ is bijective and to take advantage of this. Moreover, even if just a part of the constructors used in the pattern are in a bijective relationship with the target language – or if only a few patterns, among those instantiated by the language, are bijective – it is possible to take advantage of the affected knowledge in the target language (the transformation is not anymore complete, but at least it is correct).

With pattern languages, it seems desirable to decompose the language L in two parts: $\hat{L}$ and $\check{L}$ such that $\mu(p) = \hat{L}$ and $\check{L} = L - \hat{L}$. We then have $L = \hat{L} \vee \check{L} \ll \hat{L} \ll p$. But no non-trivial results are currently available about such a decomposition.

Like in the family of language approach, the mappings could be used for refining patterns themselves (i.e. $\mu : \mathcal{P} \longrightarrow \mathcal{L} \cup \mathcal{P}$). Then, as before, meet and join among patterns can be defined and a lattices of patterns can be extracted from which the frontier between bijective and non-bijective mappings for a particular language L can be extracted and systematically exploited.

6 Semantic description and transformations

As seen above, the expression of semantic interoperability relies on two ingredients: τ and $\models$. Its expression in machine-readable form can be achieved in various ways. τ can be expressed in XSLT or some similar language, but nothing really practical has been set up for $\models$.

We have defined the notion of Document Semantic Description (DSD) which enables to describe the formal semantics of an XML language (just like the DTD or schemas express the syntax). The DSD language, defined in XML takes advantage of Xpath for expressing references to sub-expressions and MathML for expressing the mathematical gear. The DLML family of languages contains the DSD of all the covered operators and is able to build automatically from the description of a logic the DSD of that logic.

DSD can be used for many purposes:

documenting language semantics for the user or the application developer who will require a precise knowledge of the semantics of constructs. This is eased by a transformation from DSD to LaTeX.

computing interpretations from the input of the base assignment of the variables.

checking proof of transformations is a very promising application in the line of the "web of trust" idea [Berners-Lee, 1998].

proving transformations in an assisted or automatic way;

inferring transformations from the semantics description is a very hard problem. However, from a given proof, it can be a straightforward task.

This program is rather ambitious. However, in some very restricted setting, this can be quite easy to set up. As an example, one can take the DLML context. Here, the languages have the same syntactic structure and the semantics of the operators remains the same across languages. Consider the $\mathcal{ALC}$ and $\mathcal{ALUE}$ languages which are known to be equivalent. The proof of equivalence is a demonstration that any operator missing in one language can be expressed in the other language (preserving interpretations). This iterative proof can

[2] $L\overline{\overline{\nabla}}L'$ and $L\overline{\overline{\wedge}}L'$ are defined as sets of languages and not as a particular language. If we want to define a lattice of language from these operators, they must be grouped in congruence classes (modulo equiexpressivity, e.g. $\mathcal{ALC}$ and $\mathcal{ALUE}$ are in the same class). But we cannot always guarantee that the result is a lattice.

be expressed, by a human being, that way:

$$\begin{aligned}
&\forall\langle D, I\rangle, \ldots \\
I((\texttt{not Nothing})) &\Leftrightarrow I(\texttt{Anything}) \\
I((\texttt{not } c)) &\Leftrightarrow I((\texttt{anot } c)) \qquad \text{for } c \in \mathcal{N}_C \\
I((\texttt{not } (\texttt{anot } c))) &\Leftrightarrow I(c) \\
I((\texttt{not } (\texttt{all } r\ c))) &\Leftrightarrow I((\texttt{csome } r\ (\texttt{not } c))) \\
&\ldots
\end{aligned}$$

It is straightforward to transform that proof into the following XSLT templates:

```
...
<xsl:template mode="process-not" match="dl:NOTHING">
  <dl:ANYTHING/>
</xsl:template>

<xsl:template mode="process-not" match="dl:CATOM">
  <dl:ANOT>
    <xsl:apply-templates select="."/>
  </dl:ANOT>
</xsl:template>

<xsl:template mode="process-not" match="dl:ANOT">
  <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template mode="process-not" match="dl:ALL">
  <dl:CSOME>
    <xsl:apply-templates select="*[1]"/>
    <xsl:apply-templates mode="process-not"
                         select="*[2]"/>
  </dl:CSOME>
</xsl:template>
...
```

The last rule tells that when encountering a `ALL` in the scope of a negation (`mode="process-not"`), it must be transformed in a `CSOME` with the non-negated transformation of the first argument and the negated transformation of the second one.

Moreover, if we use a language for describing proofs in conjunction with DSD, then it is possible to document the languages with DSD, the transformation with the proof and a client application will have everything that is required for proof-checking the transformation before using it.

This shows that this approach can be useful in the context of family of languages. It can be useful in the context of the ontology pattern too. Again, having the proof of the semantic preservation of μ^{-1} is the key to having a correct transformation from L to L'.

7 Conclusion and discussion

The semantic web could be a distributed web of knowledge structure and interoperability can be problematic when knowledge is expressed in different languages or in function of different ontologies. This will be an obstacle to taking advantage of imported knowledge. Semantic interoperability attempts to ensure that the interpretation of imported and transformed knowledge remains the same across languages.

We have presented a framework for expressing semantic interoperability based on the notion of transformations and semantic consequences. It has been used to analyze the various techniques employed in order to enforce interoperability. Because it is a common framework, it can be used in order to articulate the various proposals and compose all the solutions into a global one. We showed that, applied to restricted settings, it helps a lot.

This work is only a preliminary study of the relation between our expression of semantic interoperability and the implemented tools for that purpose. It seems clear, however, that many refinements are possible in the context of particular families of languages or restrictions of knowledge patterns for formally ensuring interoperability.

One of our goal is the construction of transformations satisfying semantic interoperability by composing more elementary transformations satisfying it. This will depend on the kind of property satisfied by the transformations and the kind of composition. If the transformations and their properties are published in the semantic web, then it becomes possible to create such compound transformations more conveniently.

It is worth recalling, that semantic interoperability is not total interoperability and that even with semantic properties there can be other interesting properties than full-fledged correctness and completeness. We hope that future work will enable to characterize precisely the expected properties in semantic terms.

References

[Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, (5), 2001. http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.

[Berners-Lee, 1998] Tim Berners-Lee. Semantic web roadmap, 1998. http://www.w3.org/DesignIssues/Semantic.html.

[Chalupsky, 2000] Hans Chalupsky. OntoMorph: a translation system for symbolic knowledge. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US)*, pages 471–482, 2000.

[Ciocoiu and Nau, 2000] Mihai Ciocoiu and Dana Nau. Ontology-based semantics. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US)*, pages 539–546, 2000. http://www.cs.umd.edu/ nau/papers/KR-2000.pdf.

[Clark *et al.*, 2000] Peter Clark, John Thompson, and Bruce Porter. Knowledge patterns. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US)*, pages 591–600, 2000.

[Donini *et al.*, 1994] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: from subsumption to instance checking. *Journal of logic and computation*, 4(4):423–452, 1994.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. The MIT press, Cambridge (MA US), 1995.

[James Clark (ed.), 1999] James Clark (ed.). XSL transformations (XSLT) version 1.0. Recommendation, W3C, 1999. http://www.w3.org/TR/xslt.

[Masolo, 2000] Claudio Masolo. *Criteri di confronto e costruzione di teorie assiomatiche per la rappresentazione della conoscenza: ontologie dello spazio e del tempo*. Tesi di dottorato, Università di Padova, Padova (IT), 2000.

[Nebel, 1990] Bernhard Nebel. *Reasoning and revision in hybrid representation systems*. Lecture Notes in Artificial Intelligence 422. Springer Verlag, Berlin (DE), 1990.

[Rutledge *et al.*, 2000] Lloyd Rutledge, Jim Davis, Jacco van Ossenbruggen, and Lynda Hardman. Inter-dimensionnal hypermedia communicative devices for rhetorical structure. In *Proceedings of the Multimedia Modeling conference, ?, (?)*, 2000. http://www.cwi.nl/ lynda/publications.html.

[Staab *et al.*, 2000] Steffen Staab, Michael Erdmann, and Alexander Mädche. Semantic patterns, 2000. http://www.aifb.uni-karlsruhe.de/ sst/Research/Publications/semanticpatterns.pdf.

[Stuckenschmidt and Visser, 2000] Heiner Stuckenschmidt and Ubbo Visser. Semantic translation based on approximate re-classification. In *Proceedings of the KR workshop on semantic approximation granularity and vagueness, Breckenridge, (CO US)*, 2000. http://www.tzi.de/ heiner/public/ApproximateReClassification.ps.gz.

[Stuckenschmidt, 2000] Heiner Stuckenschmidt. A pattern-based ontology language, 2000. in preparation.

[Visser *et al.*, 2000] Ubbo Visser, Heiner Stuckenschmidt, G. Schuster, and Thomas Vögele. Ontologies for geographic information processing. 2000. http://www.tzi.de/buster/papers/Ontologies.pdf.

Understanding top-level ontological distinctions

Aldo Gangemi[1], Nicola Guarino[2], Claudio Masolo[3], Alessandro Oltramari[2]

1. ITBM-CNR, Rome, Italy
2. LADSEB-CNR, Padova, Italy
(3) Dept. of Electronics and Computer Science, University of Padova, Italy
nicola.guarino@ladseb.pd.cnr.it

Abstract

The main goal of this paper is to present a systematic methodology for selecting general ontological categories to be used for multiple practical purposes. After a brief overview of our basic assumptions concerning the way a useful top-level ontology should be linked to language and cognition, we present a set of primitive relations that we believe play a foundational role. On the basis of these relations, we define a few formal properties, which combined together help to understand and clarify the nature of many common ontological distinctions.

1 Introduction

1.1 Goals of this paper

The main goal of this paper is to present a systematic methodology for selecting general ontological categories to be used for multiple practical purposes. After a brief overview of our basic assumptions concerning the way a useful top-level ontology should be linked to language and cognition, we present a set of formal (i.e., domain-neutral) primitive relations that we believe play a foundational role. On the basis of these relations, we define a few further properties, which combined together help to understand and clarify the nature of many common ontological distinctions.

Our attempt is to avoid strong ontological commitments in the early steps of the methodology, trying first to establish the formal framework needed to understand, compare, and evaluate the ontological choices that ultimately will be taken.

1.2 Limits of this paper

We are conscious that our task is very ambitious, as it necessarily faces deep and highly debated philosophical and technical problems. So we have tried to be as humble as possible, making drastic simplifications whenever possible, but trying however to save the logical rigor.

One of the most serious simplifications we have made concerns the treatment of time, which is not addressed explicitly. This is in part because we believe that ontological choices about time need to be taken after a more general ontological framework is established and in part just because temporal issues are hard.

2 Ontology, cognition and language

Is ontology about the "real world" (as seen, say, by a physicist)? Or, rather, should it take cognition into account, including the complex interactions and dependencies between our ecological niche and us? We will not attempt a general answer to this question, but we believe that the latter position is very useful when building ontologies for practical purposes. As knowledge systems manage information relevant to human agents, their ontologies need to make room for entities that depend on our perception and language, and ultimately on the way we use reality to survive and communicate. Some of these entities will depend on specific groups of human beings (with their own culture, language, and so on); others will reflect common cognitive structures that depend on our sensorial interaction with reality. A general-purpose ontology is specially interested to the latter kind of entities, which help generalize our specific knowledge of the world. This position reflects the so-called "interactionist" paradigm, which (though not prevalent) has strong support in psychology of perception and cognitive linguistics [Gibson 1977, Lakoff and Johnson 1999] and seems to be a good compromise between hard 'referentialist' ontology and purely context-oriented semiotics.

An extreme example of how ontologically relevant entities depend on our perceptive and cognitive structures is the notion of *constellation*: is a constellation a genuine thing, different from a collection of stars? This is not so clear at a first sight. But, if we distinguish between stars and their specific arrangements, we are able to understand how constellations may be considered as cognitive *things* dependent on states of mind. To see a "Taurus" in the sky does not mean, obviously, that an animal is flying in the space or (less obviously) that a bull-shaped astronomic object (different from a collection of stars) is localized in a region of the sky. Rather, the perspective we embrace consists in recognizing a *cognitive entity* dependent on the way we perceive some particular arrangement of stars. It is to this entity that we (often) refer when we use the term "Taurus constellation" in our language. Including cognitive entities in our ontol-

ogy seems therefore a good idea if natural language plays a relevant role in our applications.

For these reasons, we believe that a very useful and important requirement for top-level ontologies is the possibility of mapping them into large lexical databases (as, for example, WordNet [Fellbaum, 1998]). Although these large lexicons present many problems and limitations, they provide i) a source for distinctions used by humans as cognitive agents; ii) a way to give understandable names to ontological entities; iii) a practical hook towards NLP applications.

But, how do lexicons and ontologies link to each other? Assuming all lexical concepts as distinct (as for example the WordNet's *synsets*), ontologies that want to have the same conceptual coverage have to contain at least every concept of a lexicon. Adding non-lexicalized nodes to ontologies would be useful under different points of view. First of all, their presence may result in a better taxonomic organization; second, they may simplify the alignment with other ontologies and lexical sources, isolating the differences and the integration problems. So, an important (though idealistic) requirement to be satisfied would be that each term of a lexicon has an unique correspondent category in the ontology and that each ontological concept maps into at most one lexical concepts.

3 Methodology and basic assumptions

The requirement of a link with language and cognition makes the task of designing a good top-level ontology even more complicated. We outline here the methodology we suggest to accomplish such a task.

3.1 The role of formal relations

In philosophy we find a distinction between *formal ontology* and *material ontology*. Intuitively, this distinction seems to deal with the "level of generality" of ontological properties and relations, but its logical implications are not very clear. Smith gives the following "definition":

> "As formal logic deals with properties of inferences which are formal in the sense they apply to inferences in virtue of their form alone, so formal ontology deals with properties of objects which are formal in the sense that they can be exemplified, in principle, by objects in all material spheres or domains of reality." [Smith, 1998]

In this sense, we can consider *formal relations* as relations involving entities in all "material spheres", so that they are understandable *per se* as a universal notions. On the contrary, *material relations* are specific to one or more material spheres. This account seems however to presuppose an *a priori* division of the domain into "material spheres": first we establish a set of primitive subdomains (categories?), and then we distinguish between formal and material relations on the basis of their scope's behavior with respect to these subdomains. So, formal relations establish the connections and the differences between primitive subdomains; while material relations characterize, in a more detailed way, the properties of a specific subdomain. If we assume a flat domain, with no *a priori* structure, then the proposed distinction between formal and material relations collapses.

In both cases, choosing the right primitives is not easy. In one case, we must answer the question: "Which are the primitive subdomains?". In the other case, the question is: "Which are the primitive relations?". The two questions look indeed quite similar.

In this work we prefer to start with a set of primitive relations defined on a flat domain, and use them to *reconstruct* the classic categorial distinctions. This choice is mainly a matter of methodological clarity and economy, and is also motivated by our desire to maintain ontological neutrality as much as possible. These primitive relations will be still called "formal", as they will be selected among those considered as "formal" in the philosophical literature. By means of these formal relations we shall be able to:

- Formulate general constraints (e.g., atomicity) on all domain entities;
- Induce distinctions between entities (e.g., dependent vs. independent), and impose a general structure on the domain.

3.2 The methodology in a nutshell[2]

The methodology we have adopted can be summed up as follows:

1. Select from the classical philosophical repertoire a set of *formal relations* (neutral with respect to the domain choice) which shall play a *foundational role* in our ontology.
2. Select and adapt from the literature the *ground* axioms for these relations, such as those concerning their algebraic properties.
3. Add *non-ground* axioms, which establish constraints across basic relations.
4. Define a set of *formal properties* induced by the formal relations.
5. Analyze systematically the allowed combinations of formal properties, introducing a set of *basic categories*
6. Classify the relevant kinds of domain entities according to the basic categories. The result will help to understand the minimal domain structure.
7. Study the dependencies/interrelationships among basic categories, introducing *intercategorial relations.*
8. Increase the depth level of ontological analysis, by iterating this methodology within each basic category.

This work is still in progress, and in this paper we discuss in detail only points 1-4 of the methodology. However, we

[1] Clarifying the distinction between so-called 3-d and 4-d ontologies is out of the purpose of this paper. We just point out that 3-d ontologists believe in a crisp distinction between objects and events (or *continuants* and *occurrents*, roughly corresponding to nouns and verbs), while 4-d ontologists don't, as they see concrete entities in terms of spatiotemporal regions.

[2] See ([Thomasson, 1999] p.111-134) for an interesting discussion which advocates a methodology for ontological analysis very similar to the present one.

hope in the possibility of a progressive methodological refinement and adjustment in the way we have outlined. Moreover, we must make clear that, to start the above process, we need first some minimal assumptions (or at least intuitions) about our *largest* domain of interest (which depend on the choices discussed in section 2). We also need to make some preliminary choices concerning the formal treatment of existence, modality, space and time. We shall not discuss these issues here, although we believe that these choices can be better understood, refined, or modified, by applying the methodology above.

4 Formal Relations

4.1 Instantiation and Membership

In the ontological engineering community, classical first order logic with equality is generally adopted as a formalization language (more or less reduced in its expressivity if computational efficiency is important). This means that we take for granted the distinction between properties and domain entities: the latter (syntactically denoted by constants) are usually called *instances* of the formers (syntactically denoted by predicates). The instantiation relation seems to have therefore an intrinsic meta-logical nature, as it links together entities belonging to different logical levels. Things are complicated by the fact that, given a theory A, we can construct a meta-level theory B whose constant symbols correspond to A's predicates, and whose intended domain is that of A's properties. So the term "instance" is ontologically ambiguous, unless the corresponding level is specified. There is however a bottom level, that of *ultimate instances*, things that cannot be predicated of anything else. These are what philosophers call 'particulars', i.e., entities that cannot be instantiated, as opposed to 'universals', i.e. entities that can be predicated on particulars[3].

Despite its apparent simplicity, the notion of instantiation is subtle, and should not be confused with that of set *membership*. Let's try to clarify this by means of a classical example. There are two possible interpretations of the sentence "Socrates is a man":

1. Socrates belongs to the class of all human beings;
2. Socrates exhibits the property of *being a man*;

Usually, in mathematics, the two views are assumed to be equivalent, and a predicate is taken as coinciding with the set of entities that satisfy it. This view is however too simplistic, since in Tarskian semantics set membership is taken as a basis to decide the *truth value* of property instantiation, so the former notion is independent from the latter. The existence of a mapping between the two relations does not justify their identification: one thing is a set, another thing is a property common to the elements of a set. A set may have many common properties, or maybe none[4]. A set has a cardinality, while a property abstracts from cardinality. A set is not something that can be multiply "present" in different things like a property: a set is a *particular*, a property is a *universal*. Membership involves the former, instantiation the latter.

So properties (universals) *correspond* to sets (called their *extension*), but are not sets. We may wonder however whether two universals that correspond to the same set are the same. Those who take intensionality into account usually refuse this assumption. The classical "realist example" of intensionality is that of the three predicates "human", "featherless biped", and "animal that laughs" that have the same extension but are considered to be different. An interesting alternative, suggested in [Lewis, 1983], is to include in the extension of a predicate all its possible instances (*possibilia*). In this case, "featherless biped" would include other instances besides humans, so that we can more safely assume that two universals are the same if they have the same extension.

A final problem concerns the possibility of having a (first order) logical theory of universals. In general, this appears to be impossible, since the predicates used to talk about universals (like instantiation) would themselves refer to universals. The solution we adopt is to reserve the term "universal" to those properties and relations whose instances are particulars. Limiting our domain to the first two levels, we can aim at building a separate meta-theory that accounts for the distinctions we need for our purposes. To stick to first order logic, however, we need to avoid quantifying on arbitrary universals. To this purpose, we adopt a practical suggestion proposed by Pat Hayes[5], to further restrict the universals we quantify on to a pre-defined set of relevant properties and relations, corresponding to the predicates explicitly mentioned in our object-level theory. Within this theory, we can state some minimal ground axioms for the instantiation relation, and introduce definitions for particulars and universals. Reading $I(x, y)$ as "x is an instance of y", we have:

(I1) $I(x, y) \rightarrow \neg I(y, x)$ (*asymmetry*)
(I2) $(I(x,y) \wedge I(x,z)) \rightarrow (\neg I(y,z) \wedge \neg I(z,y))$ (*antitransitivity*)

$Par(x) \triangleq \neg\exists y(I(y, x))$
$Uni(x) \triangleq \neg Par(x)$

We shall not discuss distinctions among universals in detail here. A preliminary discussion on this topic (focused on properties) has been published in [Guarino and Welty, 2000a].

4.2 Parthood

The *parthood* relation is a very basic and investigated notion, which has been formalized only at the beginning of 20th century [Leonard and Goodman, 1940; Lesniewski, 1991]. These works intend to build a single theory (called

[3] The term "universal" is due to the fact that, metaphorically, we may see a property as multiply *present* in different things. Note however that this doesn't mean that different instances of the same universal have any *part* in common.

[4] In other words, a set doesn't coincide with its characteristic function.

[5] Message to the IEEE SUO list, http://suo.ieee.org

classical extensional mereology) that, unlike set theory, is founded only on concrete entities. More recently, [Simons, 1987] and [Casati and Varzi, 1999] pointed out that we can have different mereologies corresponding to different parthood relations, and made explicit the formal dependencies among them.

We shall write $P(x, y)$ as "*x is a part of y*". Only three ground axioms (P1-P3) are considered as minimal, although the *weak supplementation* axiom (P4) is often accepted:

(P1) $P(x, x)$
(P2) $(P(x, y) \wedge P(y, x)) \rightarrow x = y$
(P3) $(P(x, y) \wedge P(y, z)) \rightarrow P(x, z)$
(P4) $PP(x, y) \rightarrow \exists z(P(z, y) \wedge \neg O(z, x))$

where

(DPP) $PP(x, y) \triangleq (P(x, y) \wedge \neg P(y, x))$
(DO) $O(x, y) \triangleq \exists z(P(z, x) \wedge P(z, y))$.

The *extensionality* axiom (P5) and the stronger *supplementation* axiom (P6) [6]:

(P5) $(\exists z(PP(z, x)) \wedge \forall z(PP(z, x) \rightarrow PP(z, y))) \rightarrow P(x, y)$
(P6) $\neg P(x, y) \rightarrow \exists z(P(z, x) \wedge \neg O(z, y))$

are much more controversial. It is safer therefore to assume they hold only for some classes of entities called *extensional*[7] entities.

Axioms guaranteeing existence of sum, difference, product, fusion of entities or establishing mereological properties (such as atomicity or divisibility) are debatable, and then we shall not commit to them at this stage of our methodology.

4.3 Connection

Parthood only is not enough to analyze the *internal structure* of a given entity, as it only allows us to check whether it is atomic or divisible. To the purpose of capturing at least some basic intuitions related to the notion of whole, connection is usually introduced in the meretopological literature [Simons, 1987; Varzi, 1999] as a further primitive in addition to parthood. It is assumed to satisfy the following minimal axioms:

Ground axioms:
(C1) $C(x, x)$
(C2) $C(x, y) \rightarrow C(y, x)$

Link with part relation:
(C3) $P(x, y) \rightarrow \forall z(C(z, x) \rightarrow C(z, y))$

Note that from (C3) and (P1) we can deduce (C2), then (C2) is redundant. The converse of (C3) is controversial, as it seems to be acceptable only for spatial regions.

These axioms can be specialized in various ways to account for different notions of connection. For instance, within topological connection between 3-d regions, it may be useful to distinguish among point-connection, line-connection, and surface-connection [Borgo *et al.*, 1996].

[6] Note that P6 implies P5, but not viceversa.
[7] Unfortunately, this adjective is used with different meanings in the literature, see section 5.2.

4.4 Location and Extension

Recently, Casati and Varzi [Casati and Varzi, 1999] have axiomatized the notion of location by means of a primitive L intended to capture the intuition of "being (exactly) in a place". Although their approach is focused on space only, we believe it can be generalized to account for the relationship existing between arbitrary entities and four-dimensional regions. Since – at least at this point – we want to be neutral about the commitments on the distinction between continuants and occurrents, we prefer renaming this relation in terms of *being extended in a (n-dimensional) region*. We introduce therefore a binary primitive $E(x, y)$ to be read as "*x* is the extension of *y*"[8], and we assume for it the axioms from [Casati and Varzi, 1999]:

Ground Axioms:
(E1) $(E(x, y) \wedge E(z,y)) \rightarrow x = z$ (*functionality*)
(E2) $E(x, y) \rightarrow E(x, x)$ (*conditional reflexivity*)

Links with parthood:
(E3) $(P(x, y) \wedge E(z, x) \wedge E(w, y)) \rightarrow P(z, w)$
(E4) $(P(x, y) \wedge E(y, z)) \rightarrow PE(x, z)$

where

$PE(x, y) \triangleq \exists z(P(z, y) \wedge E(x, z))$ (*partial extension*)

Link with connection:

(E5) $(C(x, y) \wedge E(z, x) \wedge E(w, y)) \rightarrow C(z, w)$

Note that transitivity and antisymmetry follow from ground axioms. Note that we do not exclude that different entities can have the same extension, and that we assume a region as something that is extended in itself. Some useful definitions follow:

$Reg(x) \triangleq E(x, x)$ (*x is a region*)
$Extd(x) \triangleq \exists y(E(y, x))$ (*x is extended*)
$Coext(x, y) \triangleq \exists z,u(E(z, x) \wedge E(u, y) \wedge u = z)$
(*x and y are co-extensional*)
$OC_\phi(x, y) \triangleq \phi(x) \wedge E(y, x) \wedge \forall z((\phi(z) \wedge E(y, z)) \rightarrow O(z, x))$
(*x* ϕ*-occupies y*)

We take for granted the further axioms introduced by [Casati and Varzi, 1999] to ensure the topological properties of regions (pp. 122-126), which will not be discussed here. On the basis of this theory, the following relevant theorem can be proved for any extensional property ϕ (see section 5.2):

$(OC_\phi(x, y) \wedge OC_\phi(z, y)) \rightarrow z = x$.

We can't prove the same for co-extensionality, and this makes clear the difference between being extended in a region and occupying that region.

4.5 Dependence

We consider here ontological dependence as a general relation potentially involving all the entities of the domain. In

[8] Note that we reversed the arguments of Casati and Varzi's L primitive.

this sense we try to understand dependence as a formal relation with a minimal ontological commitment, which can be specialized in different ways. The classical philosophical reference for the notion of dependence is Husserl's work [Husserl, 1970]. Recently, Fine and Simons [Simons, 1987; Fine, 1995b] have suggested some alternative formalizations of Husserl's analysis, discussing their problems and possible solutions.

Following Husserl, Fine proposes four axioms for dependence:

Ground Axioms:
(D1) $D(x, x)$
(D2) $D(x, y) \wedge D(y, z) \rightarrow D(x, z)$

Links with Part relation:
(D3) $P(x, y) \rightarrow D(y, x)$
(D4) $\exists y(D(x, y) \wedge \forall z(D(x, z) \rightarrow P(z, y)))$

Is this a good axiomatization of dependence relation? A minor problem is that (D1) is provable from (D3) and from the reflexivity of parthood, (P1), then (D1) is redundant. Another problem regards (D4). This axiom guarantees the existence of an entity *y* that is the maximal (with respect to part relation) entity from which *x* depends, and then it is clearly not ontologically neutral.

Moreover, Simons criticizes these axioms from a more general point of view. He points out that these axioms can be interpreted in terms of weak topological structures, where dependent entities correspond to non-closed sets, and independent entities correspond to closed sets. Dependence would therefore resemble a sort of topological relation, and this may sound as counterintuitive.

Indeed, ontological dependence is usually not introduced as a primitive relation, but rather defined in terms of a modal operator and an existence predicate (*Ex*) as:

(DD) $D(x, y) \triangleq \Box(Ex(x) \rightarrow Ex(y))$.

In order to accept this, we have to accept however that dependence is intrinsically linked to modality, and somebody finds this debatable, too. If we want to be neutral with respect to this issue, we need a theory that is compatible with the modal interpretation of *D* relation. But, as Simons points out, if we interpret *D* as in (DD), axiom (D3) is satisfied only if we either subscribe to mereological essentialism (any part of *x* is necessarily such) or if we consider a modal interpretation of *P* (part means *essential part*). Otherwise in general from "*x* is part of *y* it does not follow that *y* could not exist without *x*" (Simons, p. 317). This is a big problem. If we abandon axioms (D3- D4) the characterization of dependence relation is really weak.

We are tempted therefore to accept (DD). In this case we have however other problems. One problem is that we may have different kinds of modal operators each inducing different kinds of dependence relations and that present technical difficulties (for example, *formal necessity*, *material necessity*, *nomological necessity*, etc.). A more serious problem is the characterization of predicate *Ex*. This seems really not so simple. For example, does *being something* coincide with *existing*? Do things like ordinary objects and events exist in the same way? (see [Fine, 1995a]). In order to clarify these issues we may introduce time too, but in this case we need either to introduce another modal operator that interacts with the first one, or to treat time independently. The latter approach has been adopted in [Thomasson, 1999], where the author informally introduces different kinds of temporal dependence, such as:

$CD(x, y) \triangleq \Box(Ex(x, t) \rightarrow Ex(y, t))$ (*constant dep.*)
$HD(x, y) \triangleq \Box(Ex(x, t) \rightarrow (Ex(y, t') \wedge t' \leq t)$ (*historical dep.*)

Thomasson also includes the possibility for a universal to depend on a specific particular (being a wife of Henry VIII depends on Henry VIII), and for a particular to depend only *generically* on another particular that instantiates a specific universal (the US generically depend on some US citizen).

We find these definitions extremely interesting intuitively, but we do not attempt at formalizing them here. So, for the time being, we take only axioms (D1) and (D2), leaving the interpretation of ontological dependence to intuition. We introduce however some useful definitions based on *P* and *D*:

$MD(x, y) \triangleq D(x, y) \wedge D(y, x)$ (*mutual dependence*)
$SD(x, y) \triangleq D(x, y) \wedge \neg D(y, x)$ (*one-side dependence*)
$ED(x, y) \triangleq D(x, y) \wedge \neg P(y, x)$ (*external dependence*)

5 Formal Properties

On the basis of the formal relations discussed above, let us briefly introduce a set of formal properties that we believe especially useful for our purposes. For the sake of simplicity, our domain of quantification will be limited to particulars so that the formal properties will not correspond to logical definitions, but will be stated in the meta-language. Those meta-level definitions that classify a particular with respect to the universal denoted by ϕ are expressed by using a ϕ subscript.

5.1 Concreteness and abstractness

In section 4.4 we have already defined the notion of an extended entity as something that extends in a (spatiotemporal) region. We shall take the property of being extended as synonymous of *being concrete.* A non-extended entity will be called *abstract*.

Note that this sense of "abstract" has nothing to do with the process of abstracting a common property from a set of entities. So the decision whether properties (or universals) are abstract or concrete, according to our terminology, cannot be taken on the basis of the theory of extension we have introduced. What the theory tells us is that, if the elements of a set or the instances of a property are concrete, we assume universals to be concrete, then we have to interpret the meaning of parthood and connection for universals in a suitable way. Moreover, we have to establish a link between the extension of a particular and the extension of the universal that it instantiates. Similar difficulties would occur assuming that sets are concrete, since in this case we

need a theory that links their extension to that of their members (and to the parts of their members). For these reasons, it seems pretty safe to stick to the usual assumption that universals and sets are both abstract. Collections, which will be discussed below, are the concrete correspondent of sets.

5.2 Extensionality

We say that an entity is *extensional* if and only if everything that has the same proper parts is identical to it:

$Extl(x) \triangleq \exists z(PP(z, x) \wedge (\forall z(PP(z, x) \leftrightarrow PP(z, y)) \rightarrow x = y)$

Examples of extensional entities are regions and amounts of matter.

We say that a property is *extensional* iff all its instances are extensional. We say in this case that this property *carries an extensional criterion of identity*[9].

Unfortunately, the adjective "extensional" is also used with different meanings in the literature. Sets are said to be extensional since they are identical when they have the same members, and properties are considered as extensional when properties with the same instances are taken as identical.

5.3 Unity and plurality

We believe that the formal relations we have introduced allow us to exactly define the notion of unity, but this requires some care.

Let us first give some definitions based on the parthood relation, which may capture some notions related to that of unity:

$At(x) \triangleq \neg\exists y(PP(y, x))$ (*atomicity*)
$Div(x) \triangleq \neg At(x)$ (*divisibility*)
$At_\phi(x) \triangleq \phi(x) \wedge \neg\exists y(\phi(y) \wedge PP(y, x))$ (ϕ-*atomicity*)
$Div_\phi(x) \triangleq \phi(x) \wedge \exists y(\phi(y) \wedge PP(y, x))$ (ϕ-*divisibility*)
$IHom_\phi(x) \triangleq \phi(x) \wedge \forall y(PP(y, x) \rightarrow \phi(y))$ (ϕ-*int. homogeneity*)
$Max_\phi(x) \triangleq \phi(x) \wedge \neg\exists y(\phi(y) \wedge PP(x, y))$ (ϕ *maximality*)
$EHom_\phi(x) \triangleq \phi(x) \wedge \forall y(PP(x, y) \rightarrow \phi(y))$($\phi$-*ext. homogeneity*)
$\Sigma_\phi(x) \triangleq \forall y(P(y, x) \rightarrow \exists z(\phi(z) \wedge P(z, x) \wedge O(z, y))$ (*sum of* ϕ*s*)

The notion of maximality seems indeed very much related to unity (wrt a certain ϕ), but it does not account for the way the various parts of *x* are bound together. Indeed, there are different aspects behind the notion of unity of an object, which are merged together in the following definition:

> "Every member of some division of the object stands in a certain relation to every other member, and no member bears this relation to anything other than members of the division." ([Simons, 1987], p.328)

We see here at least three fundamental aspects: a notion of division within a whole, with *members* of such division; a suitable *unifying relation* that binds the members together, and a *maximality constraint* with respect to this relation on the members. The notion of "member of a division" is the subtle issue here. Simons takes class membership as a primitive distinct from parthood. We'd rather analyze it in terms of parthood, in the spirit of the analysis presented in [Guarino and Welty, 2000b]. The definition of unity proposed in that paper has however some problems[10], so we propose here a new one that captures more carefully the notion of member of a division.

Our intuition is that any member of *x* is a *special part* of *x*. So we need a property that individuates the members within the parts of *x*. Observe that, if *x* forms a unity under a unifying relation *R*, then the property we need must only pick up the parts of *x* that belong to *R*'s domain. All other parts can be ignored[11].

Note now that *R* must be at least symmetric and reflexive (let's postpone by now the discussion about transitivity). Then we can define a predicate δ_R denoting *R's* domain, which must hold when *x* is a member of a division unified by *R:*

$\delta_R(x) \triangleq R(x, x)$

We can observe that, if *R* is defined on the whole domain, then $\delta_R(y)$ also holds for all the parts *y* of *x*.

We define now the notion of a whole as follows:

$\upsilon_R(x) \triangleq \Sigma\delta_R(x) \wedge \forall y,z((\delta_R(y) \wedge \delta_R(z) \wedge P(y, x) \wedge P(z, x)) \rightarrow R(y, z))$ (*x is unified by R*)
$\omega_R(x) \triangleq Max_{\upsilon_R}(x)$ (*x is a whole under R*)

The first definition says that *x* is unified by *R* iff it is a sum of entities belonging to *R*'s domain, and all these entities are linked together by *R*. The second one says that *x* is a whole under *R* iff it is *maximally* unified by *R*.

Let us discuss now the assumptions regarding *R*'s transitivity. At a first sight, it would be obvious to assume *R* as transitive; together with the previous assumptions, this would result in *R* being an equivalence relation. However, this would exclude the possibility of *overlapping wholes* with a *common* unifying relation[12]. Consider for example the notions of *committee* or *organization*: two committees may have a member in common while being two different wholes. Of course, in a strict sense, there would two different unifying relations in this case (say, having mission *A* vs. having mission *B*). The point is that there would be no *common* unifying relation attached to the property *committee*. A plausible common relation would be "having the same mission", but this is not transitive. This is why, con-

[9] An extensive analysis of criteria of identity has been done elsewhere [Guarino and Welty, 2000b; Guarino and Welty, 2001]; for our purposes, we shall only distinguish here between extensional and non-extensional identity criteria.

[10] Consider the following counterexample: suppose you want to say that all the children *a, b, c* of a certain person form a whole. So all the parts of *a+b+c* must be linked together by the unifying relation "having the same parent". But two of them, namely a+b and b+c, are not linked by such relation, since they are not persons. Another problem is linked to the fact that the previous definition excludes the possibility of overlapping of entities that are wholes (see below).

[11] We may also study the property of *internal uniformity* of *x* with respect the predicate $\phi(y) \triangleq R(y, x)$ but this is another problem.

[12] We are grateful to Aaron Kaplan for this counterexample.

trary to the previous papers by Guarino and Welty, we shall not assume transitivity for *R*.

For the purpose of ontological analysis, it is interesting to explore how various unifying relations can be defined on the basis of simpler relations, called *characteristic relations* ([Simons, 1987] p.330). This analysis can be used to introduce different kinds of wholes.

In particular, from the cognitive point of view, it is very interesting to consider topological connection as a characteristic relation. More exactly, the cognitively relevant characteristic relations are those that restrict topological connection to hold between physical entities of the same kind, such as matter, color, or physical bodies (otherwise, using topological connection only, the only whole would be the universe).

Under this perspective, take C_φ as the transitive closure of the projection of C on φ entities. We say that x is a *topological whole under* φ if $\omega_{C_\varphi}(x)$.

We can now introduce the notions of *singularity* and *plurality*, assuming that they are cognitively bound to topological connection:

$Sing_\varphi(x) \triangleq \omega_{C_\varphi}(x)$ (*singularity*)
$Plur_\varphi(x) \triangleq \neg Sing_\varphi(x) \wedge \exists y(PP(y, x) \wedge Sing_\varphi(y))$ (*plurality*)

A singular entity is therefore one that is a topological whole. We define a plurality as anything that contains a topological whole and is not itself a topological whole.

Topological wholes have two parameters, corresponding to the C and φ above. If we take C as the usual topological connection, an isolated piece of matter will be a topological whole under "matter", while a spot of color will be a topological whole under "color". Note that nothing excludes a topological whole (under a certain kind of connection) to include other topological wholes (under a different kind of connection): think of a lump of spheres, which can be seen as a whole under point connection and contains many wholes under surface connection.

Note that if something does not contain a whole, it will be neither singular nor plural (think for instance of an undetached piece of matter).

A special case of plurality is a *collection*, which must be a sum of wholes. Each of these wholes will be a member of the collection.

Within singular entities, it may be interesting to distinguish between homogenous and non-homogeneous entities with respect to ϕ:

$Simple_{\varphi\phi}(x) \triangleq Sing_\varphi(x) \wedge IHom_\phi(x)$
$Complex_{\varphi\phi}(x) \triangleq Sing_\varphi(x) \wedge \neg IHom_\phi(x)$

For instance, if we assume singularity as based on point connection (that is, roughly, physical contact), we have that a physical body is homogenous wrt surface-self-connection, while an assembly formed by different bodies that touch each others is not.

5.4 Dependence and Independence

Finally, the last formal property that we consider is whether or not an entity is *externally* dependent, i.e. dependent on other things besides its parts:

$Dep(x) \triangleq \exists y(ED(x, y))$ (*dependence*)
$Dep_\phi(x) \triangleq \exists y(\phi(y) \wedge ED(x, y))$ (ϕ-*dep., or generic dep.*)
$Ind(x) \triangleq \neg Dep(x)$ (*independence*)

According to our discussion in section 4.5, we have however to select an intended interpretation for *D*, since there are different kinds of dependence. For the purpose of isolating broad, relevant categories of entities, we believe that a special importance should be given to what Thomasson calls *constant dependence*, whose proper formalization requires an account of time that must be subject of future work. Under this view, for example, we can stipulate that ordinary objects (*continuants*) are independent, while events (*occurrents*) are dependent. More work on this is needed, however.

Conclusions

Developing a well-founded top-level ontology is an very difficult task, that requires a carefully designed methodology and rigorous formal framework. We hope to have contributed on both these aspects.

Since this is work in progress, we haven't been able to explore and discuss in detail the practical consequences of the methodology we have presented, although we have definite evidence of its relevance.

We are presently at step 4 of the sequence discussed in section 3.2. We hope in the possibility of a cooperative effort to proceed (through refinements and adjustments) in the way we have outlined.

This work has been done in the framework of the European Eureka project E!2235 "IKF" (Intelligent Knowledge Fusion). In this framework, we plan to develop a general reference ontology linked to a lexical resource such as WordNet, by using the methodology we have outlined. The final result will be of public domain, and will hopefully profit from (and contribute to) existing cooperation initiatives in this area, such as the IEEE SUO.

Bibliography

[Borgo *et al.*, 1996] Stefano Borgo, Nicola Guarino and Claudio Masolo. A Pointless Theory of Space based on Strong Connection and Congruence. In *Principles of Knowledge Representation and Reasoning (KR96)*, pp. 220-229, Boston, MA, 1996.

[Carrol, 1956] J. B. Carrol (ed.). *Language, Thought and Reality: Selected Writings of Benjamin Lee Whorf*. MIT Press, New York, 1956.

[Casati and Varzi, 1999] Roberto Casati and Achille Varzi. *Parts and Places*. MIT Press, Cambridge, 1999.

[Fellbaum, 1998] C. Fellbaum (ed.), *WordNet. An Electronic Lexical Database*. MIT Press, Cambridge, 1998.

[Fine, 1995a] K. Fine. Ontological Dependence. *Proceedings of the Aristotelian Society*, 95:269-90,1995a.

[Fine, 1995b] K. Fine. Part-Whole. In B. Smith and D.W. Smith (eds.), *The Cambridge Companion to Husserl*, pp. 463-485, New York, Cambridge University Press, 1995b.

[Gibson, 1977] J. J. Gibson. The Theory of Affordances. In R. E. Shaw, J. Bransford (eds.), *Perceiving, Acting and Knowing*, Hillsdale, LEA, 1977.

[Guarino and Welty, 2000a] Nicola Guarino and Christopher Welty. A Formal Ontology of Properties. In *Proceedings of the ECAI-00 Workshop on Applications of Ontologies and Problem Solving Methods*, pp. 12-1 12-8, Berlin, Germany, 2000a.

[Guarino and Welty, 2000b] Nicola Guarino and Christopher Welty. Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In Werner Horn (ed.) *ECAI-2000: Proceedings of the 14th European Conference on Artificial Intelligence*, pp. 219-223, IOS Press, Berlin, Germany, 2000b.

[Guarino and Welty, 2001] Nicola Guarino and Christopher Welty. Subsumption and Identity. Technical Report n. 01/2001, LADSEB-CNR, Italy, 2001.

[Hjelmslev, 1961] Louis Hjelmslev, *Prolegomena to a Theory of Language*. Madison, University of Wisconsin Press, 1961 (translation of Omkring sprogteoriens grundlæggelse, Copenhagen, 1943)

[Husserl, 1970] Edmund Husserl. *Logical Investigations.* Routledge and Kegan Paul, London, 1970.

[Kaplan, 2001] A. N. Kaplan, *Personal Communication*. 2001.

[Lakoff and Johnson, 1999] G. Lakoff and M. Johnson. *Philosophy in the Flesh: The Embodied Mind and Its Challenge to Western Thought*. Basic Books, 1999.

[Leonard and Goodman, 1940] H. Leonard, S. and N. Goodman. The Calculus of Individuals and Its Uses. *Journal of Symbolic Logic*, 5:45-55,1940.

[Lesniewski, 1991] S. Lesniewski. *Collected Works*. Kluwer, Dordrecht, 1991.

[Lewis, 1983] David Lewis. New Work for a Theory of Universals. *Australasian Journal of Philosophy*, 61(4)1983.

[Simons, 1987] P. Simons. *Parts: a Study in Ontology.* Clarendon Press, Oxford, 1987.

[Smith, 1998] Barry Smith. Basic Concepts of Formal Ontology. In Nicola Guarino (ed.) *Formal Ontology in Information Systems*, pp. 19-28, IOS Press, Amsterdam, 1998.

[Thomasson, 1999] Amie L. Thomasson. *Fiction and Metaphysics.* Cambridge University Press, Cambridge, 1999.

---

i

Building and Exploiting Ontologies for an Automobile Project Memory

Joanna Golebiowska [1 2], Rose Dieng-Kuntz [1], Olivier Corby [1], Didier Mousseau [2]

1 INRIA, ACACIA Project, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France
2 RENAULT, TPZ D12 138, DTSI/DTPU/KMPD, sce 18820 860 quai de Stalingrad, 92109 Boulogne, France
E-mail: {Joanna.Golebiowska, Rose.Dieng, Olivier.Corby}@sophia.inria.fr

Abstract

This paper describes SAMOVAR (Systems Analysis of Modelling and Validation of Renault Automobiles), aiming at preserving and exploiting the memory of past projects in automobile design (in particular the memory of the problems encountered during a project) so as to exploit them in new projects. SAMOVAR relies on (1) the building of ontologies (in particular, thanks to the use of a linguistic tool on a textual corpus in order to enrich a core ontology in a semi-automatic way), (2) the «semantic» annotations of the descriptions of problems relatively to these ontologies, (3) the formalisation of the ontologies and annotations in RDF(S) so as to integrate in SAMOVAR the tool CORESE that enables an ontology-guided search in the base of the problem descriptions.

Keywords: Design and engineering of domain ontologies ;Ontology-based search and retrieval of information ; Knowledge management solutions for large organizations.

1 Introduction

How to preserve and exploit the memory of past projects in automobile design (in particular the memory of the problems encountered during a project) so as to exploit them in new projects? The role of ontologies for knowledge management is more and more. They can play an important role for building a project memory, that is a specific kind of corporate memory [Dieng et al, 1999, 2000]. Several researchers aim at proposing a methodology for building such ontologies, possibly from textual information sources [Aussenac-Gilles *et al*, 2000a]. Such a methodological framework is interesting for us, as there are several heterogeneous sources of information inside the company: different databases, official references, problem management systems and other specific bases in the departments; moreover, in addition to basic data which can be processed by traditional means, some bases contain important textual data.

After detailing our problematic and the concrete problem to be solved at Renault, we will present the approach adopted for SAMOVAR. Then we will detail our techniques for building the SAMOVAR ontologies, relying on both manual construction and semi-automatic construction thanks to the application of heuristic rules on the output of a linguistic tool applied on a textual corpus stemming from textual comments of a database. Then we will explain their exploitation and the use of the CORESE (Conceptual Resource Search Engine) tool [Corby et al, 2000] for information retrieval about the descriptions of past problems encountered in vehicle projects. We will generalize our approach so as to propose a method for building a project memory in the framework of any complex system design. In our conclusion, we will compare SAMOVAR to related work.

2 The problematic

The field of SAMOVAR is the process of prototype validation during a vehicle project. This process is intrinsically complex and raises many problems. These problems frequently slow down the cycle due to the necessity of repeating validations: so, it increases both the delays and the costs of such projects.

A close observation of validation shows that part of the failure is due to loss of information and of experience gained. The objective of SAMOVAR is to improve the exploitation of this information and make it available for future projects. Useful data exist in the form of text. Therefore it is necessary to find suitable techniques and tools, such as for example linguistic techniques for exploiting the knowledge underlying such texts.

2.1 Context

The product development cycle of an automobile is made of numerous repetitive sub-cycles (design / development / validation) - of short or long duration. The whole cycle is punctuated by milestones and prototype waves which mark the production of successive models and prototypes, more or less complex. During a vehicle project, validations are carried out: the testing department checks that the component-parts or the functions satisfy the requirements of the product specifications.

Thus, the quality of smoothness of the dashboard, the noise of a car door being shut, the behaviour of the car on cobble stones, or even its resistance to high or low temperatures are tested. These validations are spread throughout the vehicle project and done successively by the testing department, starting from the most elementary functions till the final synthesis test. The project begins with tests related to the engineering center according to the parts validated and ends with tests on performance, speed and crash.

These project validation phases often reveal discrepancies with respect to the specifications. From detection of a problem to its resolution, such problems are documented in a unique data management system called Problem Management System (PMS). This system uses a database including the information needed for the process of problem management: especially information on the actors involved in the project and above all, the descriptions and comments on the problems that arose.

2.2 Interest of exploiting the Problem Management System

The appearance of problems increases the additional costs and the project duration. Therefore solutions have been thought out. One possible solution would be to exploit the information contained in the PMS in order to use the PMS not only as a problem management system but also as a source of information.

The PMS can be considered as a huge source of information, thanks to the textual fields of the base which are particularly rich and under-exploited. The actors involved in the automobile design project express themselves freely for describing the problems detected, as well as the various solutions proposed, or the constraints for carrying out such or such solution. This base can therefore be considered as archives or even as constituting (a part of) the memory of a project, more precisely the memory of the problems encountered during the project.

Furthermore, in the company, there are other information sources, such as the official corporate referential or the numerous local bases of the testing department. It would be useful to exploit this information with the contents of the PMS.

Therefore our aim is to propose a means of retrieving, structuring and making reusable this wide quantity of information for the same project or for the other projects. The participants of current projects have expressed needs related to information search and retrieval useful during the validation phases. Their needs concerned especially the retrieval of similar incidents, detection of any correlation or dependency with other incidents and so the reuse of existing solutions within the same or even a different project.

Some pieces of information are relatively simple to retrieve. However, this is not the case for the textual data of PMS. The vocabulary used by the project participants in such comments is broad and varied: a given term (existing in the corporate official referential) frequently has different designations according to the department or even the phase reached in the project. Therefore, our objective was to detect a suitable semantic term, to classify it according to the validation process and to link it with all the variations encountered. So, we needed to extract the main terms of the domain (and the relations between them if possible) and to structure them in our ontology.

2.3 SAMOVAR's approach

A synthesis of tools dedicated to the extraction of terms and of relations from textual corpora is proposed in [*Aussenac et al*, 2000]. Several linguistic tools exist to extract candidate terms: Lexter [Bourigault, 1994], Nomino[1] , Ana [Enguehard, 1992] [Enguehard and Pantera, 1995]. With regard to the acquisition of semantic relations, several approaches enable to acquire them (based on the exploitation of syntactical contexts : [Grefenstette, 1994], or the use of the lexical-syntactical patterns : [Hearts, 1992], [Desclès and Jouis, 1993]). Few tools are offered such as Coatis [Garcia, 1998] for causal relationships, Cameleon [Seguela, 1999] [Seguela and Aussenac-Gilles, 1999] for hyponymy and meronymy relations.

The approach of SAMOVAR consists of structuring the knowledge contained in the PMS textual fields describing problems, and of enabling the user to carry out searches with the aim of finding similar problem-descriptions

As a starting point, we took directly the exploitable sources (i.e. the different databases of the company), and then we built up several ontologies offering different viewpoints on the validation process: problems, projects, services, components (i.e. parts). After having primed our base manually, we completed it progressively, with the elements from the PMS textual data using Natural Language Processing (NLP) tools – in particular, Nomino that was chosen as term extractor for availability reasons. This stage

[1] http://www.ling.uqam.ca/nomino

is automatic, however the support of an expert is necessary throughout the process. Then we annotated the problem descriptions automatically with instances of concepts of the ontologies. Finally we facilitated the access to the base of problem-descriptions thanks to the formalization in RDF(S) of the ontologies and of the annotations, enabling the use of the CORESE tool [Corby et al, 2000] to carry out ontology-guided searches through the such annotated base of problem-descriptions. The wholeSAMOVAR approach is summarized in figure 6.

3 SAMOVAR ontologies

The SAMOVAR base is composed of 4 ontologies, each dedicated to the description of a precise field :

- *Component* Ontology: it is based on the official company referential, corresponding to the functional segmentation of a vehicle into sub-components;
- *Problem* Ontology: it contains the problem types and it is built up semi-automatically from a manually-activated core from textual fields taken from the problem management system;
- *Service* Ontology: it corresponds to the services cross-referenced with the company organization (management and profession) and it is supplemented by PMS information. This ontology gives an additional overall point of view on the problems;
- *Project* Ontology: it reflects the structure of a project and it is made up of knowledge acquired during a project vehicle, according to the interviews carried out with different actors on the project.

Each ontology is a n-leveled hierarchy of concepts linked by the specialization link.

Remark: Instead of building several interconnected ontologies, we could have built one single ontology organized through several sub-ontologies. We chose to distinguish the different ontologies in order to enable their possible reuse independently from one another.

3.1 Construction of the ontologies

The ontologies were built through two phases according to the data type and the means involved:

- a first extraction of the information contained in data bases,
- a second extraction, with specific techniques and tools for discovering the information « hidden » in texts.

The core of our ontology was primed manually, thanks to elements stemming from existing bases (see figure 1).

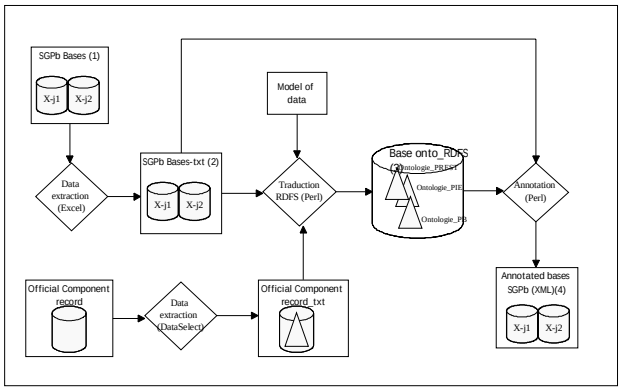


Figure 1: ***Construction of ontologies for SAMOVAR***

A first extraction of the initial data (1) supplied a textual format (2) which was then translated in the form of an ontology, by respecting the RDFS format (as expected by CORESE). In parallel, another extraction was made from the Component referential in order to complete the previous data with additional information. Then the ontological base (3) was used to annotate the data with the terms designating concepts of the ontologies. Thus we obtained the initial base annotated with annotations related to the concepts of the ontologies (4).

A second process deals with the textual data (the final goal being to enrich the result of the first extraction with the information stemming from the texts).

This process exploits the output obtained after application of the linguistic tool Nomino on the textual corpus stemming from the textual comments contained in the problem management system (PMS). Nomino is a tool for extraction of nominal groups from a representative corpus in a domain. Nomino takes as input a textual corpus and produces as output a set of « lexicons » - lists of nouns, nominal complex units (NCU), additional nominal complex units (ANCU), verbs, adjectives, adverbs. The (A)NCU corresponds to the prepositional groups (PG) or the nominal groups (NG). The lexicons of the NCU are accessible in the form of graphs which illustrate the existing dependencies for a PG or a NG.

Then, we exploited the lexicons and the graphs produced by Nomino, in order to :

- detect the significant terms (i.e. corresponding to important validation points in the automobile design validation process),
- enrich the *Problem* ontology by means of the Nomino graphs, by exploiting the regularity of their structures.

Detection of significant terms

Firstly, we analysed the lexicons produced by Nomino in order to discover the most frequent terms, likely to be the most representative terms of the domain : *wiring, assembling, pipe, attachment, centring, component, installation, conformity, branch, hole, clip, screw, contact,*

maintains, *tightening*, *paw*, *position*, *geometry*, *nut*, *to screw*, *hygiene*, *connecting*.
These structured terms allowed us to set up the *Problem* ontology. The initial structuring of this ontology was based on discussions with the experts. Figure 2 shows an extract of this *Problem* ontology.

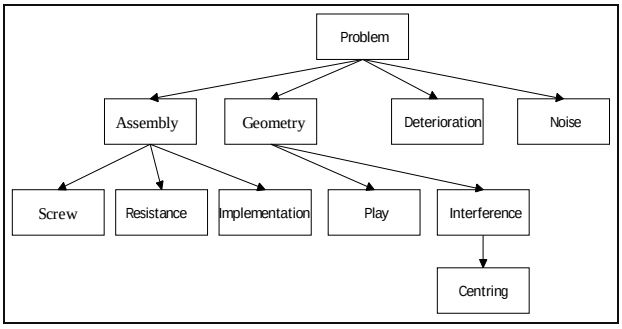


Figure 2: ***Extract of the Problem Ontology***

The terms selected for the bootstrap were those which are exploitable as semantic clues for a problem type: for example, a problem of `Centring` can be discovered thanks to the presence of such clues as «`indexage`», `coaxiality`, «`entraxe`», etc.
Indeed the Nomino outputs can be sorted by frequence numbers. The most frequent words can be considered as relevant fr the processed domain and we exploit them as clues for the *Problem* ontology bootstrap.
The validity of the terms (i.e. the candidate terms for the bootstrap, and the clues exploited to find them) was confirmed with support of the experts.

Once the bootstrap of ontology was constituted, it needed to be enriched. For this purpose, we used the prepositional groups stemming from Nomino.

Enrichment of the *Problem* ontology
Besides nouns, Nomino produces nominal and prepositional groups. We exploited the structures of the most frequent cases : figure 3 shows an extract of the ANCU produced by Nomino.

```
difficulte_de_mep, difficulte_de_mise_ne_place,
difficulte_d'alignement, difficulte_de_chaussage,
difficulte_d'emmanchement, difficulte_d'accostage,
difficulte_d'agraphage, difficulte_d'acces,
difficulte_de_vissage, difficulte_de_serrage,
difficulte_de_emmanchement, difficulte_de_montage,
durete_de_clipsage, durete_de_connexion,
durete_de_manoeuvre, durete_de_mep,
probleme_de_clipsage, clipsage_impossible,
clipsage_difficile clipsage_inefficace,
probleme_de_fixation, eclairage_insuffisant,
effort_de_clipsage, effort_de_declipsage,
effort_de_mep, effort_de_montage,
effort_de_positionnement, effort_de_raccordement,
effort_de_serrage, effort_de_sertissage,
effort_de_sertissage_insuffisant,
effort_d_encliquetage, effort_de_branchement,
effort_de_chaussage, collier_agressif,
deplacement_goulotte, deplacement_locating,
deterioration_de_connecteur,
deterioriation_lecheur_ext,
detrompage_insuffisant, gene_pour_clipsage,
gene_pour_fixation,
gene_pour_la_fixation_du_presseur,
gene_pour_la_mep,
gene_pour_la_mep_de_l_agrafe_du_cablage_moteur,
gene_pour_la_mep_de_l_agrafe_tuyau,
gene_pour_la_mep_du_monogramme,
gene_pour_la_mep_du_rivet,
gene_pour_la_mep_du_tuyau_hp,
impossibilite_de_clipsage_du_tuyau,
impossibilite_de_mep_du_protecteur,
impossibilite_de_montage_de_la_facade_de_console,
impossibilite_de_mep_boitier_gps/gsm,
mal_indexee_sur_moteur,
mal_placee_pour_l_operateur,
mal_positionnees_sur_cablage, manque_boutonniere,
mauvais_tenu_du_gicleur, mauvais_tenu_du_moteur,
mauvais_centrage, mep_difficile
```

Figure 3: ***Extract of ANCU produced by Nomino***

The manual analysis of these NCU was performed by studying each Nomino output carefully so as to find some regularities in the NCU obtained by Nomino. This manual analysis, carried out with the support of the expert, supplied the structures which we exploited to build the SAMOVAR heuristic rules. For instance, we could find cases such as:

(*DIFFICULTE EFFORT PROBLEME DURETE MANQUE RISQUE EFFORT*) DE PROBLEME
GENE POUR PROBLEME DE PIECE
MAUVAIS PROBLEME DE PIECE
IMPOSSIBILITE DE PROBLEME DE PIECE
PROBLEME(*INCORRECT IMPOSSIBLE INSUFFISANT DIFFICILE*)
(*DETERIORATION DEPLACEMENT MANQUE RUPTURE CASSE*) DE PIECE
PIECE(*DETERIORE AGRESSIF INEFFICACE*)

or in English:

(*DIFFICULTY EFFORT PROBLEM HARDNESS LACK RISK EFFORT*) OF PROBLEM
DISCOMFORT FOR PROBLEM OF PART
BAD PROBLEM OF PART
IMPOSSIBILITY OF PROBLEM OF PART
PROBLEM(*INCORRECT IMPOSSIBLE INSUFFICIENT DIFFICULT*)
(*DAMAGE DISPLACEMENT LACK BREAK BREAKAGE*) OF PART
PART (*DAMAGED AGRESSIVE INEFFICIENT*)

We exploited these structural regularities of Nomino outputs to build manually heuristics rules validated by the expert, heuristic rules which would enable the feeding of the ontology in a semi-automatic way.
These rules that reflected the existing structures in the corpus were determined manually, but once implemented and activated, they helped us to enrich the *Problem* ontology automatically by suggesting to attach a relevant new concept corresponding to a new term, at the right position in the ontology. Figure 4 shows examples of heuristic rules.

```
R1 : Noun [type=Problem,n=i]
Prep[lemma=« of »]
Noun[type=Problem,n=i+1] ;
R2 :(difficulty||effort||hardness||lack||risk||effort) Prep[lemma=« of »]
Noun[type=Problem]
R3 : impossibility Prep[lemma=« of »]
Noun[type=Problem]  Prep[lemme=« of »]
Noun[type=Component]
R4 : Noun[type=Problem]
Prep[lemma=« of »||lemma=« on »||lemma=« under »] Noun[type=Component]
```

Figure 4: ***Examples of heuristic rules***

These rules represent the possible combinations between the elements of the *Component* and *Problem* ontologies as attested in the texts. A rule is presented as a series of categories, each one possibly decorated with a set of features (for example type=Problem to indicate that the element is part of the *Problem* ontology, type=Component for an element of *Component* ontology, etc.).
These rules were implemented in PERL.

Kinematic of the process

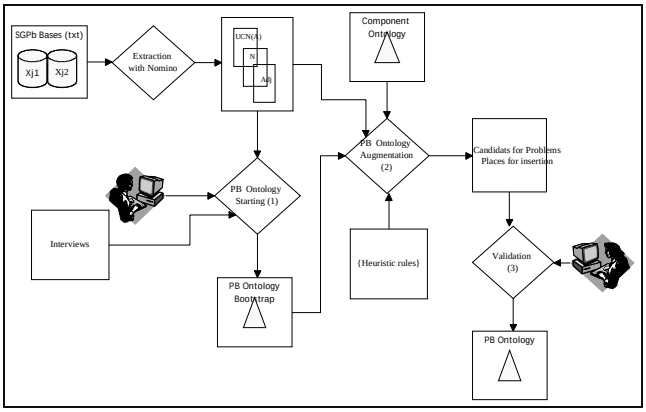


Figure 5: ***Process of enrichment of the ontology Problem***

We enriched the *Problem* ontology gradually. For that, the SAMOVAR system takes in entry the Nomino outputs, the *Component* ontology, and the heuristic rule base. Then it analyses the nominal groups to see with which rule each of them can match.

Example of a Nominal Group and the corresponding rule:

NOISE OF RUBBING OF THE WHEEL DURING ITS HEIGHT ADJUSTMENT

Noun[type=Problem,n=i] Prep[lemma=« of »]
Nom[type=Problem,n=i+1] ;[2]

The rule matches the nominal group, recognises the first term as a noise (that corresponds to an existing concept in the *Problem* ontology) and proposes to build a concept for the second noun and to insert it in the *Problem* ontology, as a son of the *Noise* concept. In the following case, the rule matches the name of the part and proposes to link the first term as a *Problem* :

JUDDERING OF THE REAR SWEEP ARM ON PPP3

Noun[type=Problem]
Prep[lemma=« of »||lemma=« on »||lemme=« under »]
Noun[type=Component] ;[3]

The output provides the candidate terms to insert in the *Problem* ontology. The knowledge engineer (possibly with the support of the expert) validates each candidate and decides if the position proposed for insertion in the existing *Problem* hierarchy is correct. If yes, a concept corresponding to the term is inserted in the ontology. Such a concept – that was attested in the textual corpus - can be compared to a «terminological concept» if we use the terminology of Terminae [Biébow and Szulman, 1999].
To formalize our ontologies, we chose the RDF Schema (RDFS) language, which is recommended by W3C for description of resources accessible by the Web. RDFS allows to simply describe the ontology to which RDF annotations will be relative to. Such RDF annotations are quite relevant to describe resources within a company. We can consider the descriptions of the problems met in a vehicle project (i.e. problem descriptions contained in PMS) as resources being a part of the memory of this project.
Therefore, we developed a parser which, at the end of the process, generates a version of the ontology in RDF Schema (which is also the formalism required by the CORESE software). After RDF(S) generation, the annotations of the PMS problem-descriptions are automatically updated by SAMOVAR in the form of RDF statements.

[2] **BRUIT DE FROTTEMENT** DU VOLANT PENDANT SON REGLAGE EN HAUTEUR

Nom[type=Problème,n=i] Prep[lemme=« de »]
Nom[type=Problème,n=i+1]

[2] **BROUTEMENT DU BRAS-BALAI AR** SUR PPP3

Nom[type=Problème]
Prep[lemme=« de »||lemme=« sur »||lemme=« sous »]
Nom[type=Pièce] ;

4 Exploitation of the Ontologies

4.1 Use of the CORESE Tool

The ontologies set up were used to make annotations on the problem-descriptions from the PMS, considered as document elements. Their formalization in RDF Schema and the formalization of the annotations in RDF enabled to use the CORESE tool for information retrieval guided by such RDF(S) ontologies and annotations [Corby *et al*, 2000].

The CORESE tool implements a RDF(S) processor based on the conceptual graph (CG) formalism [Sowa, 1984]. CORESE relies on RDF(S) to express and exchange metadata about documents. CORESE offers a query and inference mechanism based on the conceptual graph (CG) formalism. It may be compared to a search engine which enables inferences on the RDF statements by translating them into CGs.

CORESE translates the classes and properties of RDFS towards CG concept types and relation. CORESE also translates the base of RDF annotations into a base of CGs. This enables the user to ask queries to the RDF/CG base. A query is presented in the form of an RDF statement which is translated by CORESE into a query graph which is then projected on the CG base (using the projection operator available in CG formalism). The graphs results of this projection are then translated back into RDF for providing the user with the answers to his query. The projection mechanism takes into account the concept type hierarchy and the relation type hierarchy (obtained by translation of the RDF schemas).

To exploit CORESE, we formalised the SAMOVAR ontologies into RDFS. Then, we indexed the problem-descriptions of the PMS base with instances of concepts from these ontologies, while respecting the XML-based RDF syntax. After these two stages, the user could carry out information retrieval from the annotated problem-description base. The results of the user's query take into account not only the initial terms of the query but the links modeled in the different ontologies.

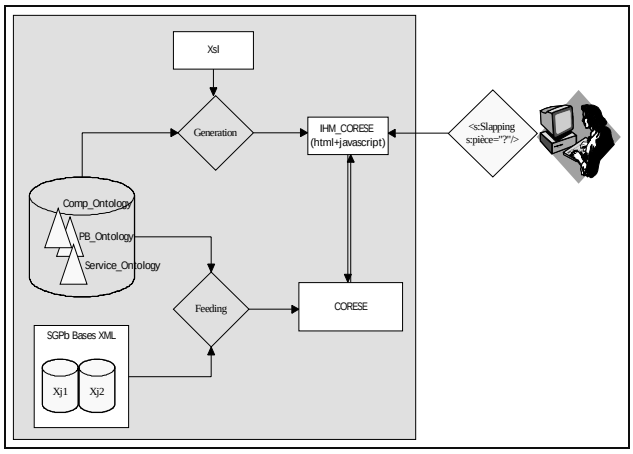


Figure 6: ***Architecture of SAMOVAR***

4.2 Examples of queries

Here are two examples in which we show that the problems extracted from texts and structured with hierarchical links allows us to find duplications of problem descriptions:

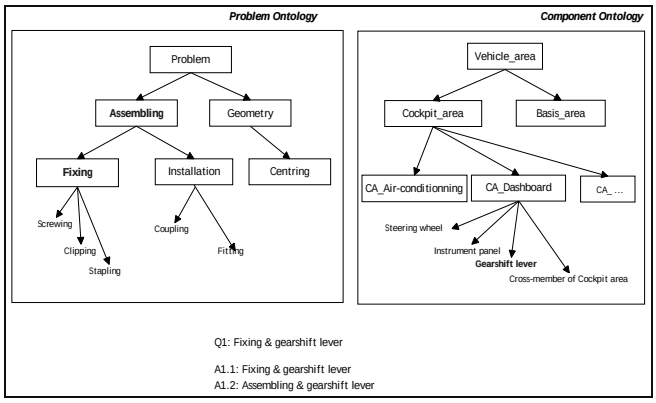


Figure 7: ***Pathway for the ontologies to retrieve information***

In the first example, the user is looking for the problems of *fixing on the gearshift lever bellows*. A single answer is obtained:

T_Fixation
rdf:about=http://coco.tpz.totto.fr:8080/SAMOVARXML/MOXj1-02057.xml

Libelle DIAMETRE DU SOUFFLET AU NIVEAU DU BOUTON PRESSION NON EN CONCORDANCE AVEC LE DIAMETRE DU POMMEAU DU SELECTEUR DE VITESSE (**VOIR PSXj2-00193**)

Piece SOUFFLET_DE_LEVIER_DE_VITESSE

On the other hand, if the user extends her query to take into account more general concepts, following the ontological links (in our case - *assembling*), she will find a second case, which is effectively a similar problem-description.
Following a successive route through the ontologies thanks to the generalization and specialization links, the user can expand the query to find the subsuming concepts (cf. the fathers of the elements of the query) and the sibling concepts. In the example, the user can explore the *problems on gearshift lever*, level by level: from problems of fixing /connecting, she can go up to the father of this last concept (i.e. *Assembling*), and then go down to the other children concepts (e.g. *Installation*). The second case thus found is a similar problem-description to the first answer :

T_Montage
rdf:about=http://coco.tpz.totto.fr:8080/SAMOVARXML/PSXj2-00193.xml

Libelle BOUTON PRESSION DU SOUFFLET DE LEVIER DE VITESSE IMMONTABLE (**GEREE PAR MOXj1-02057**)

Piece SOUFFLET_DE_LEVIER_DE_VITESSE

In the second example, the user would like to find the *problems of centring on crossbar of cockpit area*. The system returns three cases among which two turn out to be problem-descriptions pointing mutually:

T_Centrage
rdf:about="http://coco.tpz.totto.fr:8080/SAMOVARXML/MOXj1-00403.xml"

libelle FIXATIONS PDB : FIXATIONS LATERALE G ET COMPTEUR DECENTRE SUR TRAVERSE.

piece TRAVERSE_DE_POSTE_DE_CONDUITE

T_Centrage
rdf:about="http://coco.tpz.totto.fr:8080/SAMOVARXML/MOXj1-02071.xml"

libelle FIXATION : SUPPORT CARMINAT SUR TRAVERSE DECENTREE. (**VOIR PSXj2-00023**)

piece TRAVERSE_DE_POSTE_DE_CONDUITE

T_Centrage
rdf:about="http://coco.tpz.totto.fr:8080/SAMOVARXML/PSXj2-00023.xml"

libelle NON COAXIALITE DES TROUS DE FIXATION SUPPORT CALCULATEUR CARMINAT SUR TRAVERSE.(**GEREE PAR MOXj1-02071**)

piece TRAVERSE_DE_POSTE_DE_CONDUITE

The browsing through the ontology lets the user browse the whole base of problem-descriptions, following the semantic axes modeled through links in the ontologies. This browsing helps the user to find similar problem-descriptions.

4.3 Evaluation of the ontologies for the search of similar problem-descriptions

The tests were made on the *Component* and *Problem* ontologies covering the corpus corresponding to an extract of the PMS base of a vehicle-project:

- a first step was concerning a specific perimeter (*Dashboard*) for 2 milestones,
- a second step processed the entire base of the project.

We created these ontologies taking the different information sources into account (official references cross-checked with items from the problem base). In professional terms the domain corresponds to the process of *assembling*. At present the *Dashboard* perimeter contains 118 concepts and 3 relations among which 22 components within 6 architectural areas, 12 sections and 3 levels reflecting the official *Component* referential. The *Problem* ontology contains about 43 types of problems. The *Service* ontology comprises 9 services extracted automatically from the base. These ontologies have been used to annotate around 351 problem-descriptions.
The whole base contains 792 concepts and 4 relations among which 467 components are structured in the same way, but updated with a typology of 39 component managers. The *Problem* ontology contains about 75 types of problems. The *Service* ontology contains about 38 types of services retrieved from base. These ontologies have been used to annotate around *4483* problem-descriptions.

Discussion
The first exploratory investigations on search of similar problem-descriptions have been proved to be interesting. All problem-descriptions mutually pointing have been found (in the case where problem-descriptions belong to the covered perimeter). Furthermore, there were less answers, but only the relevant ones.
So, we can conclude that good results are obtained thanks to the annotations of problem-descriptions with the instances of the problem types discovered from texts and structured in an ontology.
We can also notice that the modeling of the ontology is essential in this method. Test modifications in the *Problem* ontology had more or less positive repercussions on the results. It is important to make sure of the validity of the ontology with the experts' support.
More generally, the method strongly depends on the corpus of the handled domain : if we reuse it for another domain, it will probably be necessary to update the heuristic rules allowing extraction of new concepts in order to cover the structures not processed. Indeed, the heuristic rules depend on the regularities found among the candidate terms extracted from the corpus.

Other « adjustments » were necessary during the process. For example, annotations with problems are at present performed by pattern matching : an annotation with a specific problem is activated as soon as the presence of some clues (for example `Centring` will be detected thanks to the presence of such clues as `indexage`, `coaxiality`, `entraxe`). According to the order of triggering of the rules, a problem-description can be annotated with instances of different ontology concepts. It would be interesting to order the rule triggering.
Besides, some other NLP tools (such as relation extractors [Garcia, 1998] [Seguela, 1999]) could help to refine furthermore the results of the *Problem* ontology construction.

As a further work, we intend to apply the same approach for building a *Solution* ontology (that would be connected to the *Problem* ontology). The same approach can be adopted: i.e. write heuristic rules from the manual analysis of the regularities of the candidate terms produced by Nomino and expressing possible solutions to the problems.
It would enable to index the problem-descriptions not only with instances of the concepts of the ontologies *Problem*, *Project*, *Service* and *Component*, but also with adequate instances of concepts of this *Solution* ontology.

6 Conclusions

6.1 Related Work

We have previously evoked several linguistic tools, dedicated to the extraction of terms and of relations from textual corpora. Among such tools, the choice of Nomino was due to both its relevance for our purposes and its availability. SAMOVAR can be compared to several approaches or tools integrating linguistic tools for extraction of candidate terms from a textual corpus.
Terminae [Biébow and Szulman, 1999] offers a methodology and an environment for building ontologies thanks to linguistic-based techniques of textual corpus analysis. The method is based on a study of the occurrences of terms in a corpus in order to extract the conceptual definitions and the environment helps the user in her modeling task by checking the characteristics of a new concept and by proposing potential family knot. Lexiclass [Assadi, 1998] offers an interesting approach for building a regional ontology from technical documents. This tool enables the classification of syntagms extracted from a corpus, in order to help the knowledge engineer to discover important conceptual fields in the domain. Lexiclass coupled with Lexter, carries out a syntagm classification from Lexter according to the terminological context of the terms.
[Aussenac-Gilles *et al*, 2000] describes a general method for building an ontology, method based on analysis of textual corpus using linguistic tools. The authors give the example of the Th(IC)2 project where they combine several tools for processing the textual corpus, each tool dedicated to a specific task (Lexter for terms extraction, Cameleon for relations, Terminae - for concept hierarchy construction) Our method is situated in such a methodological framework: we use various specific tools in every step of the process, but with a corpus stemming from different origins (i.e. both interviews and textual data retrieved from existing databases). This variety characterizes the originality of our approach. [Maedche and Staab, 2000a, 2000b] [Kietz *et al*, 2000] also present a general architecture for building an ontology from a textual corpus. [Maedche and Staab, 2000a, 2000b] exploit different linguistic tools so as to build a concept taxonomy and exploit a learning algorithm for mining non-taxonomic relations from texts.
The integration of CORESE in SAMOVAR and its ability to enable information retrieval thanks to annotations linked to the concepts of the ontologies thus build in a semi-automatic way is one originality of SAMOVAR. We must notice that SAMOVAR thus implements an approach for finding similar problems among past problem descriptions, which is a typical capability of case-based reasoning systems [Moussavi, 1999].

6.2 Further work

As noticed earlier, we will study heuristic rules for extraction of the *Solution* ontology from the textual corpus. Moreover, making explicit the links between the *Problem* and the *Solution* ontologies would enable to refine the indexing of the problem descriptions. Therefore, we will exploit a linguistic tool enabling the extraction of domain-dependent semantic relations, adapted to the automobile domain.

6.3 Towards a Method for Building a Project Memory

By finding information about similar problems processed during a given project, SAMOVAR has begun the process of capitalization in the company. It would be henceforth possible to spread it to wider scale - to exploit the incidents and the existing solutions between different vehicle projects, to study problems and solutions within the same range or the same project, and in longer term, exploit this capitalization to discover the recurring problems of a company by making re-show tender spots "generators of problems " in the engineering centres. So SAMOVAR could enhance information sharing among the teams involved in the same or different vehicle projects.

We could exploit the SAMOVAR principles for other projects, provided that the right adaptations are carried out, especially at the level of the ontologies. We can thus generalize our approach to other domains than automobile design, for example to build and exploit a memory of the

project of design or construction of any complex system, particularly regarding the memory of the problems encountered in such projects (e.g. incidents met during the design of a plane, a satellite, even a power plant, etc.). We propose a method relying on the following steps:

1. If there exists a database or a referential describing the components of this complex system, exploit it to build semi-automatically a *Component* ontology. Otherwise, use linguistic tools and method such as the ones described in [Aussenac-Gilles *et al*, 2000b] in order to build this Component ontology.
2. If there exists a description of a project characteristics in the considered company, exploit it to build a *Project* ontology. Otherwise, rely on interviews of the experts.
3. Establish a corpus of texts describing the problems met during one or several existing projects. It can involve texts resulting from textual documents or from textual comments in databases.
4. Exploit some existing linguistic tools allowing the extraction of candidate terms (e.g. Lexter [Bourigault, 1994, 1996] or Nomino for French texts).
5. Analyse manually (with the support of an expert) the regularities among the candidate terms which are liable to describe types of problems (resp. solutions). Then thanks to the regularities observed, write heuristic rules exploiting both these regularities and the *Component* and *Project* ontologies in order to suggest terms to include as concepts into the *Problem* (resp. *Solution*) ontology and even more to propose their position in this ontology. Validate such heuristic rules by the expert.
6. Use these heuristic rules and let an expert validate the propositions of the system obtained thanks to these heuristic rules.
7. Use the concepts of the *Problem*, *Solution*, *Component* and *Project* ontologies, so as to index automatically the elementary problem-descriptions (in the textual corpus) with instances of these concepts.
8. Exploit an RDFS generator for the ontologies and an RDF generator for the annotations, in order to be able to use the search engine CORESE to query the base annotated by the instances of problems.

The proposed methodology is generic. However the rules are constructed relying on the corpus: they reflect the existing structures of the corpus and are strongly connected to it. So, to apply the methodology for another domain it will be necessary to rebuild the heuristic rule base, so as to make it reflect the regularities observed in the corpus. This is typical of a methodology based on corpus analysis.

Acknowledgments
We wish to thank our colleagues for their precious advices on our work and for their contribution in reading over this article.

References

[Assadi, 1998] Assadi H, Construction of a regional ontology form text and its use with a documentary system. In N. Guarino, ed. Proc. of the 1st Int. Conf. On Formal Ontology and Information Systems (FOIS'98), p. 236-249, IOS Press, 1998.

[Aussenac-Gilles et al 2000a] Aussenac-Gilles N, Biébow B, Szulman S, Corpus analysis for conceptual modelling, EKAW'2000 Workshop Ontologies and Texts, Juan-les-Pins, October 2-6, 2000 pages 13-20

[Aussenac-Gilles et al, 2000b] Aussenac-Gilles N, Biébow B, Szulman S, Revisiting Ontology Design : a Method Based on Corpus Analysis, In R. Dieng and O. Corby eds, Knowledge Engineering and Knowledge Management: Methods, Models and Tools, EKAW 2000, Juan-les-Pins, French Riviera, October 2-6, 2000, p. 172-188.

[Biébow and Szulman, 1999] Biébow B, Szulman S, Terminae : a linguistics-based tool for building of a domain ontology, In D. Fensel and R. Studer, eds, Knowledge Acquisition, Modeling and Management, Proc. of the 11th European Workshop (EKAW'99), LNAI 1621,. Springer-Verlag, 1999.

[Bourigault, 1994] Bourigault D, Lexter, un Logiciel d'Extraction de TERminologie, Application à l'acquisition des connaissances à partir de textes, PhD thesis, E.H.E.S.S, Paris, France, 1994

[Bourigault, 1996] Bourigault D, Lexter, a natural language processing tool for terminology extraction. Proc. of the 7th EURALEX Int. Congress, Goteborg, 1996.

[Brickley, 2000] Brickley D. and Guha R.V. eds. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000, http://www.w3.org/TR/rdf-schema

[Corby et al, 2000] Corby O, Dieng R , Hébert C, A Conceptual Graph Model for W3C Resource Description Framework, ICCS'2000, Springer-Verlag, Darmstadt, August 2000.

[Dieng *et al*, 1999] Dieng R., Corby O., Giboin A. and Ribière M. Methods and Tools for Corporate Knowledge Management. In S. Decker and F. Maurer eds, International Journal of Human-Computer Studies, Special issue on Knowledge Management, 51:567-598, September 1999.

[Dieng *et al*, 2000] Dieng R., Corby O., Giboin A., Golebiowska J., Matta N. and Ribière M. Méthodes et Outils pour la Gestion des Connaissances, Dunod, 2000.

[Enguehard, 1992] Enguehard C, ANA, Apprentissage Naturel Automatique d'un réseau sémantique, thèse de doctorat, UTC, 1992

[Enguehard and Pantera, 1995] Enguehard C, and Pantera L. Automatic natural acquisition of terminology. Journal of Quantitative Linguistics, 2/1:27-32, 1995.

[Faure and Nédellec, 1999] D. Faure and C. Nédellec., In D. Fensel and R. Studer, editors, Proc. of the 11th European Workshop (EKAW'99), LNAI 1621,. Springer-Verlag, 1999.

[Garcia, 1998] Garcia D, Analyse automatique des textes pour l'organisation causale des actions. Réalisation du

système informatique COATIS, PhD thesis, Université de PARIS IV, Paris 1998

[Golebiowska, 2000a] Golebiowska J, SAMOVAR - Knowledge Capitalization in the Automobile Industry aided by Ontologies, PKAW 2000, Sydney, December 11-13, 2000.

[Golebiowska, 2000b] Golebiowska J, SAMOVAR - Setting up and Exploitation of Ontologies for capitalising on Vehicle Project Knowledge. In Aussenac-Gilles N., Biébow B., Szulman S., eds, Proc. of EKAW'2000 Workshop Ontologies and Texts, Juan-les-Pins, October 2000 pages 79-90

[Grefenstette, 1994] Grefenstette G, Explorations in automatic thesaurus discovery, Kluwer Academic Publishers, Boston, 1994

[Hearst, 1992] Hearst M, Automatic Acquisition of Hyponyms from Large Text Corpora, ICCL, COLING 92, Nantes July 25-28, 1992

[Jouis, 1993] Jouis C, Contribution à la conceptualisation et à la Modélisation des connaissances à partir d'un analyse linguistique de textes. Réalisation d'un prototype : le système SEEK. Thèse de doctorat, 1993, EHESS.

[Kietz *et al*, 2000] Kietz J.-U., Maedche A. and Volz R. A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet. In Aussenac-Gilles N., Biébow B., Szulman S., EKAW'2000 Workshop Ontologies and Texts, Juan-les-Pins, October 2-6, 2000 pages 37-50.

[Lassila, 1999] O. Lassila and R. R. Swick eds. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http ://www.w3.org/TR/REC-rdf-syntax

[Maedche and Staab, 2000a] Maedche A. and Staab S., Mining Ontologies from Texts. In Dieng R. and Corby O. eds, Knowledge Engineering and Knowledge Management: Methods, Models and Tools, EKAW 2000, Juan-les-Pins, French Riviera, October 2-6, 2000, p. 189-202.

[Maedche and Staab, 2000b] Maedche A. and Staab S., Discovering conceptual relations from text. Proc. of ECAI'2000, IOS Press, August 2000.

[Morin, 1999a] Morin E. Acquisition de patrons lexico-syntaxiques caractéristiques d'une relation sémantique, TAL (Traitement Automatique des Langues), 1999

[Morin, 1999b] Morin E Automatic acquisition of semantic relations between terms from technical corpora. Proc. of the 5th Int. Congress on Terminology and Knowledge Engineering (TKE'99), 1999.

[Moussavi, 1999] Moussavi M. - A Case-Based Approach to Knowledge Management, in Aha D.W. (Ed). Proc. of the AAAI'99 Workshop on "Exploring Synergies of Knowledge Management and Case-Based Reasoning". Juillet 1999; Orlando, FL. AAAI Press Technical Report WS-99-10.

[Séguéla and Aussenac-Gilles, 1999] Séguéla P. and Aussenac-Gilles N.. Extraction de relations sémantiques entre termes et enrichissement de modèles du domaine. IC'99, pages 79-88, Paris, 1999.

[Séguéla, 1999] Séguela P, Adaptation semi-automatique d'une base de marqueurs de relations sémantiques sur des corpus spécialisés. Terminologies Nouvelles, 19:52-60, 1999.

[Séguéla, 2001] Séguéla P. Construction de modèles de connaissances par analyse linguistique de relations lexicales dans les documents techniques. PhD Thesis, Université de Toulouse, March 2001.

[Sowa,1984] Sowa J. F. Conceptual Graphs : Information Processing in Mind and Machine. Reading, Addison Wesley, 1984.

myPlanet: an ontology-driven Web-based personalised news service

Yannis Kalfoglou, John Domingue, Enrico Motta,
Maria Vargas-Vera, Simon Buckingham Shum
Knowledge Media Institute(KMi),
The Open University,
Milton Keynes MK7 6AA, UK
{y.kalfoglou,j.b.domingue, e.motta, m.vargas-vera, s.buckingham.shum}@open.ac.uk

Abstract

In this paper we present *myPlanet*, an ontology-driven personalised Web-based service. We extended the existing infrastructure of the **PlanetOnto** news publishing system. Our concerns were mainly to provide lightweight means for ontology maintenance and ease the access to repositories of news items, a rich resource for information sharing. We reason about the information being shared by providing an ontology-driven interest-profiling tool which enable users to specify their interests. We also developed ontology-driven heuristics to find news items related to users' interests. This paper argues for the role of ontology-driven personalised Web-based services in information sharing.

1 Introduction

Nowadays, we observe a trend in providing personalized Web-based services in order to accommodate the versatile needs of an ever increasing number of Web users. Recent advances in agent and Internet technology provide the technological means, however, equally important is to provide the means for semantic infrastructure. Towards this goal, [Huhns and Stephens, 1999] propose the use of "personal ontologies" where each Web user will be able to create his/her own ontology tailored to his/her view of the world. Although we found this idea fruitful, it bears a contradictory connotation. When we talk about ontologies, we can't really say "personal". Ontologies are - by definition - shared views of the world([Kalfoglou, 2000a]). We rather prefer to use the metaphor "personal views" of an ontology tailored to specific services. That is, each user will see - and eventually be able to edit - part of an ontology that is tailored to a specific service. The ontology itself will remain shared, in the sense that the creation, editing and maintenance tasks involve the efforts of many agents(let them be people or software). The way it will be exposed to users will depend on the kind of services they want. For example, in our domain of Web-based news services, a user is able to browse those contents of the ontology that are related to news items, like people who wrote them, projects mentioned, etc. This kind of Web-based news services enable users to access information tailored to their interests.

Valuable information is shared among the members of a community by using the lowest-common-denominator medium: an email message. Users send a story in the form of an email(hereafter, e-Story) to a news server from which designated systems redirect the e-Story back to targeted members of the community. This is an indirect form of communication(in comparison with a member-to-member form), however, we enrich it by an ontology-driven interest-profiling tool and deductive knowledge retrieval techniques. This allowed us to reason about the knowledge being shared and target it to certain people. The means for connecting knowledge to people were analyzed from the process point of view in [O'Leary, 1998]. His framework has been used in some ontology applications([Kalfoglou, 2000b]) and in [Domingue and Motta, 2000] the authors showed how these processes are realized in the context of **PlanetOnto**. In particular they focussed on the two connecting processes: *people to knowledge* and *knowledge to people*. The means which were used to connect people to knowledge in **PlanetOnto** were integrated visualisation, search, and query-answering facilities whereas the connection of knowledge to people achieved by pro-actively contacting people to solicit e-Stories and alert them when items of interest were published.

To deliver such an ontology-driven service we need to have flexible mechanisms for ontology maintenance, an area which is still in its infancy and hampers ontology applications([Kalfoglou *et al.*, 2000]). In this work, we deployed Information Extraction(hereafter, IE) systems to extract information from users using a service which could be used to update the underlying ontology. In that sense, the user becomes the main agent responsible for maintaining the ontology instances, lifting the burder from ontological engineers who can focus on structural and semantic issues related with ontology design and deployment. In our domain we experimented with extracting information from users' e-Stories in order to update the underlying ontology.

Our research goals are two-fold: (a) to improve and ease ontology usability for Web users by means of ontology-driven Web-based front-ends to personalized services; and (b) to provide lightweight means for ontology maintenance triggered by users' input by deploying IE techniques along with domain specific templates. This tight coupling of Web-based environments with underlying ontologies is a promising and appealing technology for the majority of users.

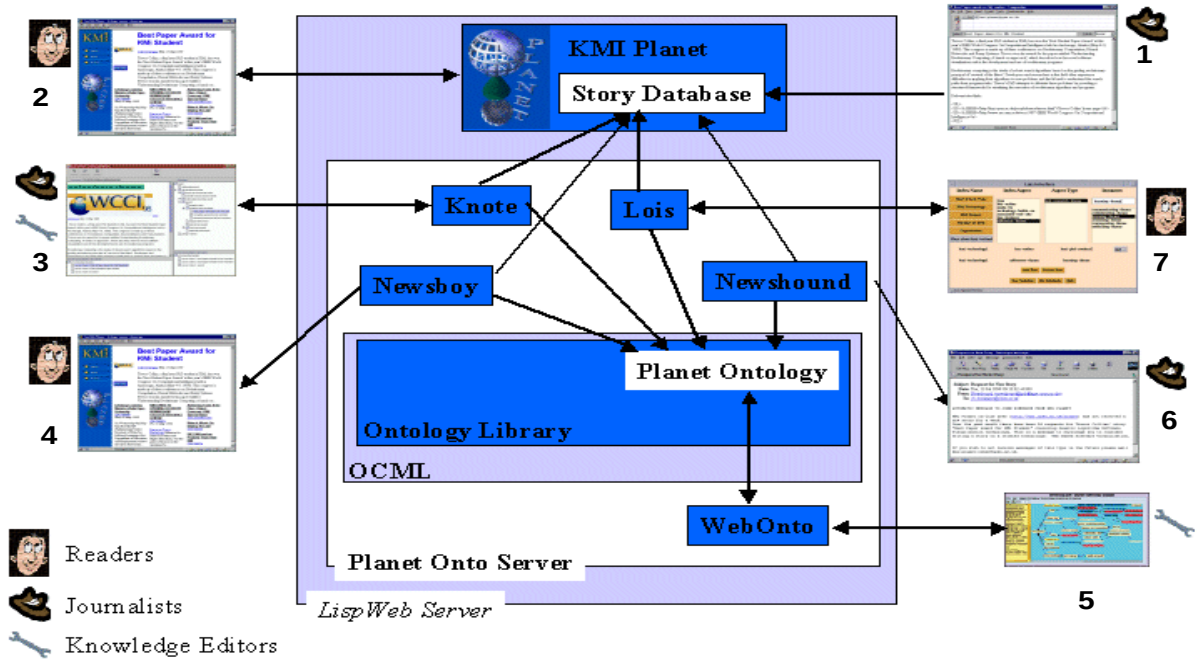


Figure 1: The **PlanetOnto** architecture.

We organize this paper as follows: in section 2 we describe the existing infrastructure, **PlanetOnto**, which we extend in section 3 with the personalized services provided by *myPlanet*. We report on related efforts in section 4 and we conclude the paper in section 5 by discussing future directions and implications of this work.

2 PlanetOnto

In this section we briefly describe the existing infrastructure, **PlanetOnto**, an integrated suite of tools developed over the last 4 years in the *Knowledge Media Institute(KMi)*. The whole infrastructure is described in detail in [Domingue and Motta, 2000]. Here we recapitulate on the important elements of the **PlanetOnto** architecture some of which were the focus of our work as we describe in the next section.

In the **PlanetOnto** domain we identify three types of users: journalists who send stories to *KMI Planet*, knowledge editors who maintain the *Planet* ontology and the *Planet* knowledge base, and readers who read the *Planet* stories. In figure 1, we illustrate the **PlanetOnto** architecture along with the activities that supports:

1. *Story submission:* Stories are submitted to *KMi Planet* in the form of email which is then formatted and stored in *KMi Planet's* story database;
2. *Story reading:* Stories can be read by using a standard Web browser;
3. *Story annotation:* A specialized tool, *KNote*, is used to help the journalist or the knowledge editor to associate the story with knowledge structure of the underlying ontology. This process was manual and we have semi-automated it as we describe in section 3.2;
4. *Provision of customized alerts:* An agent, *Newsboy*, builds user profiles from patterns of access to *PlanetOnto* and then uses these profiles to alert readers about relevant stories. While that tool uses statistical evidence to build profiles, in section 3.1 we present *myPlanet* which makes it possible for a user to build a profile by using an ontology-drawn structure;
5. *Ontology editing:* A Web-based ontology editor, *WebOnto*[Domingue, 1998], is used for constructing knowledge models in the OCML language[Motta, 1999];
6. *Story soliciting:* An agent, *Newshound*, gathers data about popular news items and then solicits potentially popular stories from the journalists. The ontology-driven heuristics of *myPlanet*, described in section 3, could extend this tool to solicit stories from journalists with similar interests;
7. *Story retrieval and query answering:* A Web-based interface, *Lois*, provides access to the story archive and the associated knowledge base by integrating Web-browsing and search with knowledge-based query retrieval.

3 *myPlanet*

PlanetOnto was originally conceived as an internal newslet-

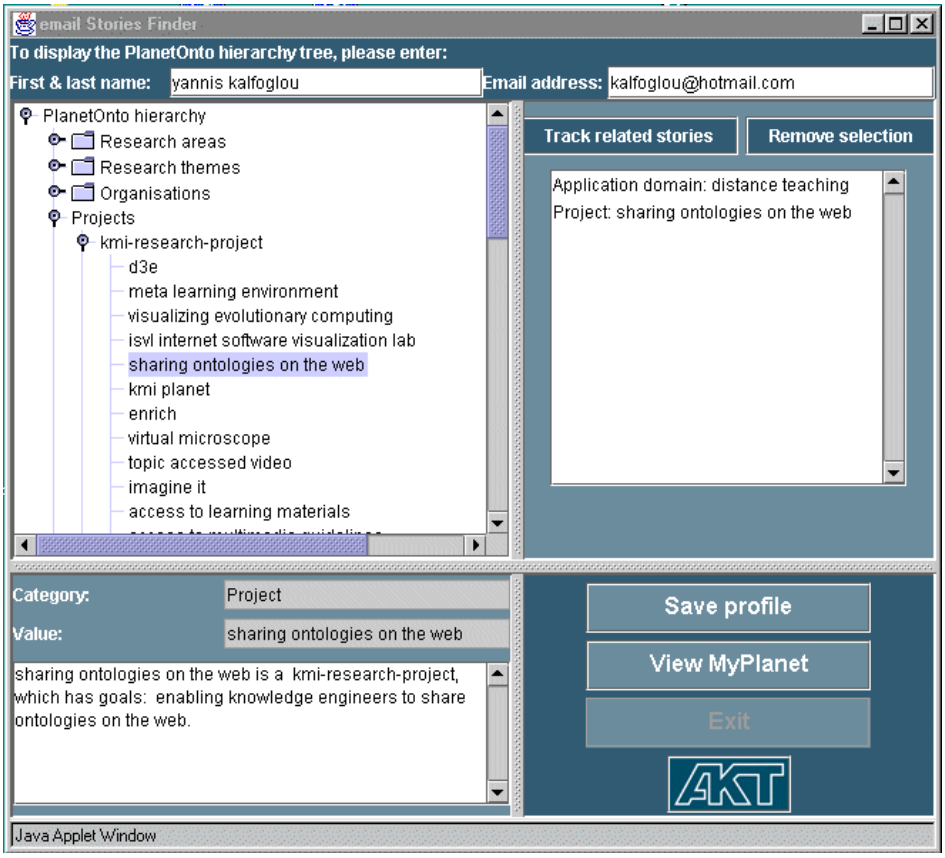


Figure 2: The e-Stories finder Java Applet.

ter and progressively became an integrated suite of tools for knowledge management. It is used as a mass communication medium from members of our lab but lacks the advantages of personalised, tailored-to-preferences, services. *myPlanet* aims to fill-in this gap by providing the means for easy navigation through the e-Stories repository, setting user preferences, and providing assistance to the knowledge editors for annotating e-Stories. We describe these tools in the following two sections.

3.1 Ontological interest-profiling

One of the limitations of the **PlanetOnto** suite of tools was the lack of an e-Stories retrieval method which would enable users to read only the e-Stories of their interest instead of forcing them to browse the e-Stories database for potentially interesting items. A possible fix to this problem would have been to provide a keyword-based search engine. This sort of solution, however, bears the known limitations that everyone of us has experienced with current keyword-based search engines(e.g, unrelated matches).

Consequently, we worked on a method which allows the user to specify his/her interests(crudely speaking, "the search criteria"), and then we search for e-Stories that match these interests. The difference of our approach when comparing it with a keyword-based search engine is that the structure of the interests is drawn from the underlying ontology. Hence, we deliberately impose a generic structure of interests to the user which contains the most important types of information one would typically find in the *KMi Planet* e-Stories. This structure is composed of the following items:

- `Research areas` that are investigated in KMi;
- `Research themes` that are investigated in KMi;
- `Organizations` that KMi collaborates with;
- `Projects` in KMi;
- `Technologies` used in KMi;
- `Application domains` that are investigated in KMi;
- `People` - members of the KMi lab.

All of these items are classes in the underlying `KMi Planet` ontology[1]. The advantage of this is that we can go beyond the expected category name matching: we can

[1]Accessible from the Web through the `WebOnto` browser on URL: http://webonto.open.ac.uk/

reason about the categories selected by applying ontology-driven deductive heuristics. For example, if someone is interested in `Research Area` *Genetic Algorithms*, we would normally return all the e-Stories that talk about that `Research Area` by employing the string-matching technique we describe in the sequel. However, by using the ontological relations that hold between these categories we can find which `Projects` have as `Research Area` *Genetic Algorithms* and then search for e-Stories that talk about these `Projects`. These would then be included in our answer set as potentially interesting e-Stories although they don't explicitly mention the *Genetic Algorithms* `Research Area`. In the same manner, we can apply more complex heuristics such as finding `Technologies` that have been used in `Projects` and `People` who are members or leaders of these `Projects` - which have as `Research Area` *Genetic Algorithms* - therefore inferring that these `People` might be a potential contact for information on `Technologies` for *Genetic Algorithms*. In terms of the underlying ontology structure, our aim is to take advantage of the rich definitions of classes in the OCML language. For example, the following OCML code is the definition of an instance of a KMi research and development project, the "sharing ontologies on the web" project:

```
(def-instance project-sharing-ontologies-on-the-web kmi-r&d-project
   ((has-research-area
    res-area-ontologies res-area-knowledge-sharing-and-reuse)
    (project-application-domain organisational-learning)
    (addresses-theme
    theme-collaborating theme-communicating theme-reasoning)
    (has-project-leader
    john-domingue enrico-motta zdenek-zdrahal)
    (funding-source org-european-commission)
    (has-goals
    "Enabling knowledge engineers to share ontologies on the web.")
    (has-web-address
    web-page-project-sharing-ontologies-on-the-web)
    (uses-technology lisp java tech-lispweb tech-ocml )
    (associated-products tech-webonto tech-tadzebao )))
```

As we can see, this definition is sufficient for deducing facts related to the project's research areas, themes, application domain, leaders, etc. Most of these constructs are used directly in the browsable structure we imposed to the user in *myPlanet*'s interface. Thus, the deduction step involves a straightforward OCML query. Other slots, however, like funding source and technologies used, can be used to infer further links as in the scenario we described before. This rich representation of a project instance highlights the strengths of OCML as a knowledge modelling language([Motta, 1999]) which has been used in many projects over the last 6 years. Currently, there are over 90 models defined in the WebOnto library all of which are accessible with a Web browser from `webonto.open.ac.uk`. We also use relations to link people with projects such as:

```
(def-relation involved-in-projects (?x ?project)
   :constraint (and (person ?x)
                    (project ?project))
   :sufficient (or (has-project-member ?project ?x)
                    (has-project-leader ?project ?X)))
```

The OCML language provides support for defining operational options for each relation such as the `:sufficient` construct in our example above. Its purpose is to help characterize the extension of a relation. For the relation given above, it is sufficient to prove that a person is a member or leader of a project in order for the relation `involved-in-project/2` to hold. We also store the selections a user makes, that is, we save the user's profile with respect to the selected interests. This profile can be edited later on as well as used for finding pro-actively e-Stories that match it.

The matching of interests in a e-Story is based on string-matching but employs the notion of "cue phrases" and "cue words" which are associated with the instances of the categories given above. We use two meanings of "cue": evidence and abstraction. A cue phrase, in our approach, is both an abstraction of the category that is associated with and evidence that the e-Story which contains it is relevant to that category. For example, we define as a cue phrase for the `Research Area` *Ontologies*, the phrase "knowledge sharing and reuse". This is an abstraction of the term *Ontologies*. Whenever we find that phrase in an e-Story we assume that this e-Story is relevant to *Ontologies*. This finding is the evidence of relevance. This technique has been proved easy to apply and gave us a broader and more accurate answer set than the one we would get with a simple match of the category name. On the other hand, we need to be careful when we identify or devise cues for a particular category since a loosely defined cue phrase could result in loosely related e-Stories. For example, the cue phrase "survival of the fittest" could be argued that is an abstraction of the *Genetic algorithms* `Research Area` since it describes a common technique of molecular biology used in *Genetic algorithms*. It might be dangerous to use it though, since it is loosely connected to the term *Genetic algorithms* and the possibility to get unrelated e-Stories is high(e.g, e-Stories about a fighting contest might contain this phrase). We see this as a tradeoff: the more generic the cue phrases are the more phrases we can define or devise, the less generic the cue phrases are the less phrases we can define or devise. It is obvious that, with more cue phrases we can find more e-Stories but the phrases can't be too generic because this may result in unrelated e-Stories. To resolve this tradeoff, we had to follow a manual approach in identifying or even devising, whenever necessary, cue phrases for all the instances of the seven categories described above. That way, we were able to judge by ourselves the "closeness" of a cue phrase to a particular category by referring to literature resources, asking experts in that category for advice, etc. We are planning, however, to automate this process to the maximum degree possible as this is a desired requirement in order to scale-up this approach in a time-effective manner.

To illustrate the usage of this tool, we will go through a detailed scenario in which a user tries to find e-Stories related to his/her interests. As we can see from figure 2, a Java Applet is used as the front-end for choosing the categories upon which the search will be based. When this Applet is loaded over the Web it loads all the instances for the seven categories given above, hence it provides a partial view of the underlying ontology's contents. In our example, the user "yannis kalfoglou" has browse the hierarchy tree and chosen two categories: `Application Domain` *Distance teaching* and `Project` *Sharing Ontologies on the Web*. These two

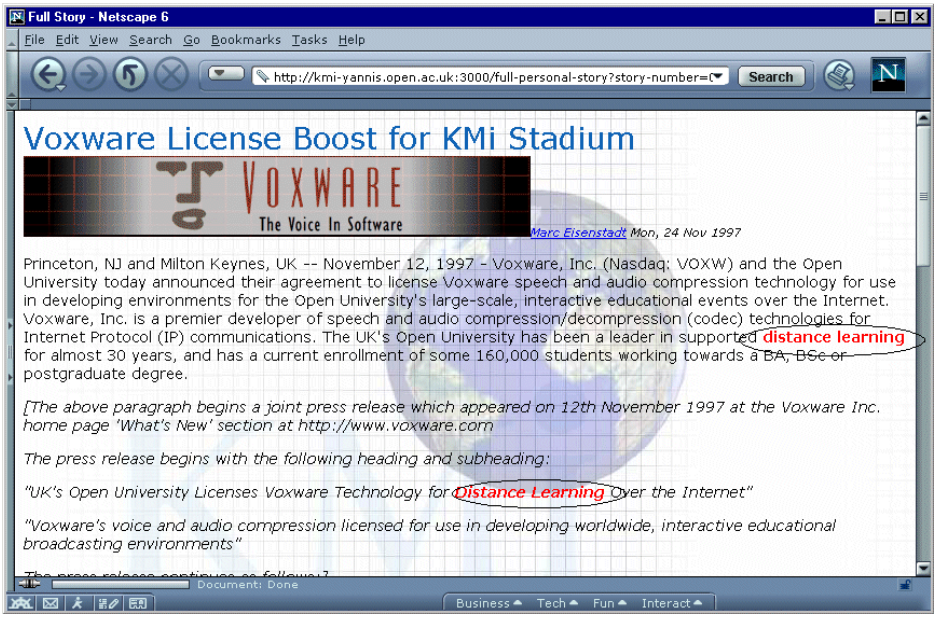


Figure 3: A e-Story of *myPlanet*.

are displayed in the upper right pane of the window in figure 2. The lower left pane is used for displaying additional information with respect to the category currently viewed in the tree. In our example, we see a textual description of the goals for the `Project` being viewed. This information is obtained by querying the underlying ontology for the project's goals. We display different types of textual information tailored to the type of category being viewed. For example, when an instance of `People` is viewed then we display the projects that this person is involved to. This information is obtained from the ontology after firing the relevant query.

After selecting the categories, user "yannis kalfoglou" can save his profile and initiate the search by pressing the `View myPlanet` button. This will display the results, if any, in a personalized Web-page which will be used in future sessions as the user's personal `Planet` Web-page(hence, *myPlanet*). Such a page contains the set of e-Stories that match the selected categories by employing the string-matching technique we described above. We include a snapshot of a e-Story that was found relevant to the user's interests in figure 3. As we can see, this e-Story contains the cue phrase "distance learning"(which is deliberately circled for the sake of this example) which is associated with the `Application domain` *Distance teaching*.

3.2 Populating the ontology

The e-Stories are formalized in terms of associating them with a formal representation which supports various forms of reasoning in **PlanetOnto**. This formalization process, as [Domingue and Motta, 2000] describe:

> "is driven by an ontology that defines the concepts needed to describe events related to academic life - for example, projects, products, seminars, publications and so forth. This means that we ignore parts of a news story that are not relevant to the ontology, much as in template-driven information extraction approaches."

In these approaches, IE systems focus only on portions of text that are relevant to a particular domain. From that perspective, IE can be seen as the task of pulling pre-defined relations from texts as we see in applications of IE in various domains(see, for example, [Proux and Chenevoy, 1997]). Furthermore, IE can be used to partially parse a piece of text in order to recognise syntactic constructs without the need of generating a complete parse tree for each sentence. This approach could be coupled with domain specific templates in order to identify relevant information. If no extraction template applies to the parsed sentence then no information is retrieved.

These characteristics of IE technology were appealing for our task: to populate the ontology with new instances of e-Stories in an automated manner. IE gave us the means to identify the part of an e-Story that will be processed, whereas domain specific templates made it possible to fill-in slots in ontology instances. For example, in a e-Story for the KMi domain one might be interested to extract only the name of KMi projects, KMi members, KMi funding organisations, KMi award bodies, money being awarded, etc., and ignore the rest. As it is described in [Vargas-Vera *et al.*, 2001], the kind of information that will be extracted is determined by the pre-defined templates which are based on the typology of events in our `KMi Planet` ontology. Examples of events are `visiting-a-place-or-people`, `academic-`

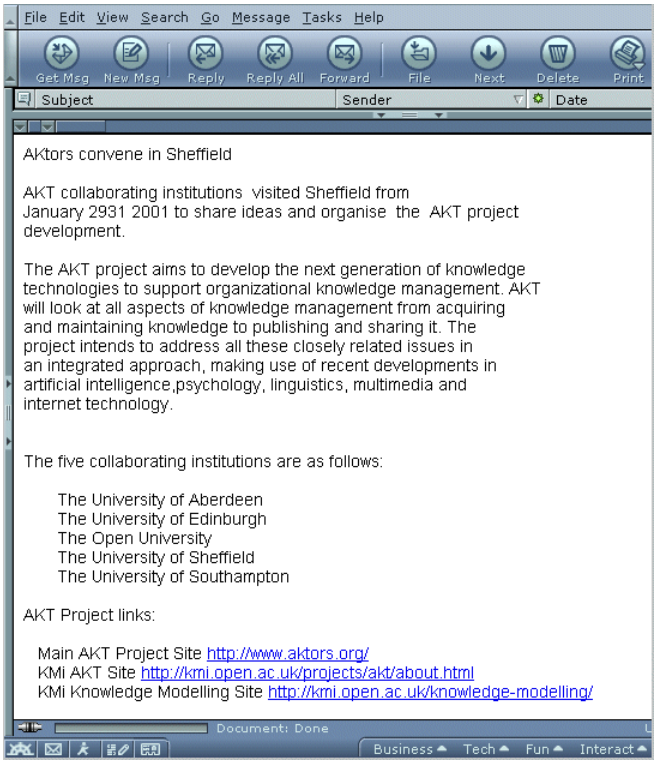


Figure 4: A e-Story send to KMi Planet.

`conference`, `event-involving-project`, and so forth. Currently, we have 40 event types defined in our ontology and we have devise templates for 10 of them. These are the domain specific templates used in IE systems.

An example template for the event type `visiting-a-place-or-people` is as follows:

`[_,X,_,visited,Y,from, Z,_]`

This template matches the sentence word list where `X` is recognisable as an entity capable of visiting, `Y` is the place being visited and cannot be a preposition, and `Z` is recognisable as a range of dates by virtue of their syntactic features. The remaining tokens in the sentence are ignored. We use the underlying `kmi-ontology` instances to identify proper names for visitors(if they are KMi employees) and whenever this fails we deploy a named entity recogniser to help us with identifying additional proper names for visitors and places. Each template is triggered by the main verb in any tense. In this template, the trigger word is the verb "visited". As [Riloff, 1996] describes, linguistic rules could be deployed to help identify trigger words reliably. For example, if the targeted information is the subject or the direct object of a verb then the trigger word should be the main verb.

Assume that a KMi journalist submits a e-Story about an AKT meeting. We illustrate such a e-Story in figure 4. As we can see, the first sentence of the e-Story matches the template given above. It contains the trigger word "visited". This will activate the template and variables `X`, `Y`, and `Z` will be instantiated to visitor, place being visited and range of dates, which give us the following information:

- visitor: "AKT collaborating institutions"
- place: "Sheffield"
- date: "January 29-31 2001"

This will be automatically converted to OCML code in order to fill-in the slots in the instance of the event type we are dealing with:

```
(def-instance visit-of-akt-collaborating-institutions
   visiting-a-place-or-people
    ((has-duration '3 days')
     (start-time january-29-2001)
     (end-time january-31-2001)
     (has-location sheffield)
     (visitor akt-collaborating-institutions)))
```

In the sequel, a form-based interface is used to visualize the information extracted as shown in figure 5. Uninstantiated slots could be filled-in manually by the knowledge engineer. The main help of this semi-automatic instantiation of event type is the extraction of information from e-Stories, the partial slots-filling, and the identification of event type.

In some templates we can also make use of the underlying ontology to support the event identification. For example, the template for the `conferring-a-monetary-award` event type is:

`[X,_,has been awarded,Y,from,Z,_]`

where `Y` is amount of money, `Z` is a funding body, and `X` is either a person of a project. To decide which one, we traverse the instances of people and projects in the underlying `kmi-ontology` to find out which matches `X`.

4 Related work

Although we couldn't find directly comparable projects with our domain - ontology-driven Web-based personalized news services - there several efforts described in the literature where ontologies and Web-based services were put together. We report on these in the sequel:

In the *FindUR* project[McGuinness, 1998], the means for knowledge-enhanced search by using ontologies were investigated. McGuinness describes a tool, deployed at the AT&T research labs, which uses ontologies to improve the search experiences from the perspectives of recall and precision as well as ease of query formation. Their tool is mainly targeted to the Information Retrieval research area and aims to improve the search engines technology. However, the idea of deploying ontologies to achieve these goals is similar to our approach which is mostly concerned with using ontologies to structure the search space(i.e., pre-selected categories of interests - section 3.1) and increase the answer set(i.e., heuristics deployed to select a relevant e-Story - section 3.1). In their work though, means for updating the topic sets used to categorize information(similar to our interests categories) were investigated. In contrast with our approach where the

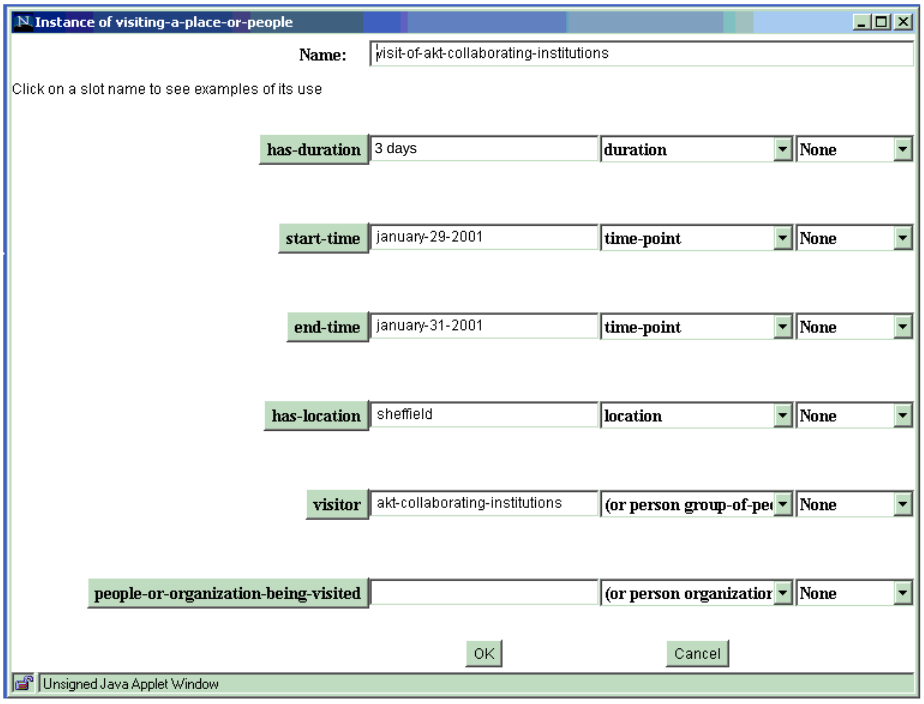


Figure 5: Semi-automatically annotate the e-Story of figure 4: a partial instantiation of the event type: *visiting a place or people*.

categories of interests are pre-defined and maintained internally, the *FindUR* team were "experimenting with a collaborative topic-building environment that allows domain experts to expand of modify topic sets directly"[McGuinness, 1998]. Although this approach has the advantage of speeding up the maintenance task, in our case we see the preselected categories as a stable piece of knowledge over time. If however, these categories need to be updated, we could use the `WebOnto`[Domingue, 1998] environment for editing and browsing the underlying ontology. We should also point out a similarity in the use of cue phrases and cue words to increase the number of related e-Stories. In the *FindUR* project, the notion of "evidence phrases" was used. However, their definition as "evidence" phrases highlights a difference in their application: as we described in section 3.1, we use cues both as abstractions of terms and as evidence whereas in the *FindUR* domain they used only as evidence. For example, as the authors describe, the company *Vocaltec* could be an evidence for the topic *Internet telephony* but certainly is not an abstraction of it. In particular, they defined a typology of evidence phrases: *synonyms*, *subclasses*, *products*, *companies*, *associated standards*, *key people*. These were then used to increase the number of related answers to a given query. They were deployed in the background along with rules that govern their interrelations. As in our approach, these were not automatically generated.

A similar approach which deploys *content matching* techniques is described in [Guarino *et al.*, 1999] where the authors present the *OntoSeek* system designed to support content-based access to the Web. As in the *FindUR* project, the target was the Information Retrieval area with the aim of improving recall and precision and the focus was two specific classes of information repositories: yellow pages and product catalogues. Their underlying mechanism uses conceptual graphs to represent queries and resources descriptions. As the authors argue, "with conceptual graphs, the problem of content matching reduces to ontology-driven graph matching, where individual nodes and arcs match if the ontology indicates that a subsumption relationship holds between them"[Guarino *et al.*, 1999]. However these graphs are not constructed auto-

matically. The *OntoSeek* team developed a semi-automatic approach in which the user has to verify the links between different nodes in the graph via a designated user-interface. The similarity of this work with *myPlanet* lies in the usage of an ontology. However, as previously, we deployed our ontology in different phases: in structuring the search space and in increasing the answer set.

On a slightly different focus, the IMPS(Internet-based Multi-agent Problem Solving) system uses software agents to conduct knowledge acquisition on-line using distributed resources[Crow and Shadbolt, 1999]. One of these agents, *OCA*(Ontology Construction Agent), is used to facilitate the task of constructing an ontology at runtime, that is, querying various resources for filling in the gaps in the ontology. Although the goals of this work were different, the underlying idea for the *OCA* is similar to our efforts of populating the ontology by automatically instantiating classes as we described in section 3.2. *OCA* was used "to extract information from networked knowledge resources - like WordNet, the online thesaurus/lexical database and a plain text domain database in the field of geology, the IGBA dataset"[Crow and Shadbolt, 1999]. Our approach is different in that we deploy IE techniques along with domain specific templates to instantiate specific ontology classes whereas the *OCA* deploys heuristic methods for extraction and focuses on creating an hierarchy lattice of classes of concepts.

In the context of managing user profiles we should point to attempts that have been made to infer user profiles from analyzing patterns of access to documents [Krulwich and Burkley, 1997]. However, most of these approaches try to induce user interests by employing empirical methods. In our case, we deliberately impose an ontology-driven structure to the user profile which enabled us to reason about it.

Finally, [Roux *et al.*, 2000] and [Faatz *et al.*, 2000] discuss early ideas on the use of IE techniques coupled with ontologies in order to help them understand complex relationships, statements or terms in semi-structured or unstructured documents.

5 Summary and future work

In this paper we presented a system, *myPlanet*, which acts as the front-end to a news server. It is placed on the top of the existing infrastructure for ontology-driven Web-based news services, **PlanetOnto**. It aims to allow users browse e-Stories according to their preferences(i.e., search criteria). The usage of the underlying ontology allowed us to devise heuristics which make it possible to increase the answer set of related e-Stories. We also provide facilities for saving users' profiles, a feature vital for providing further services tailored to their preferences.

While the ease of accessibility to our e-Stories repository was a primary goal, equally important was the maintenance of this repository. Since we base our services on the enrichment of e-Stories in terms of annotating them with ontology-drawn knowledge structures we had to find ways of automating this process. We used IE techniques and developed domain specific templates to automatically identify the event type of a e-Story and extract specific information needed for instantiating it in the underlying ontology.

There are certain research issues which remain open in this work. In the area of personalised services we need to take the ontology-based reasoning to a further stage: reason about the kind of output that will be dispatched to the user by analysing his/her profile. Since we save the user's preferences we could apply deductive heuristics to find e-Stories that are related to these preferences by means of tracing their interrelations in the underlying ontology. A simple example could be to infer that technologies used in projects might be of interest to users that looking for e-Stories related to other projects with the same research area. Furthermore, we are investigating the possibility of extending the type of output. Currently, a related e-Story is the output of *myPlanet*. In the future though, we might want to provide other kind of output like, for example, suggestions about potential collaborators on a research topic, or organizations with a potential interest in the user's research areas. These could be inferred by applying the same style of deductive heuristics but changing the output to a designated "personal interests" Web-page. As in the existing system, editing facilities are vital to keep the system updated and let the user drive the reasoning process.

One of the advantages of our "lowest-common-denominator" medium(the email message) is that we make no commitments as to what the structure should be. Which means that we can apply exactly the same infrastructure to any kind of document, not necessarily email messages. The technology needs no changes, however we might need to edit or even create new ontologies to characterise the new domain. Towards this direction, we plan to extend the usage of IE techniques coupled with domain specific templates as it has been proved a fast way of instantiating our ontologies. In our ontology population task we had to manually construct the templates for each type of event. We are planning to automate this task by deploying inductive learning algorithms. The existing set of e-Stories could be used, potentially, as the training set to identify characteristics of event types which will eventually lead to automatically construct their templates. These templates can then be tested on the annotated e-Stories to judge their quality and appropriateness. In the same line of work, we intend to expand on IE techniques and include tools that allow detection of anaphora which is an important feature when dealing with large corpusses of text from the same organisation but different departments. In these cases, terms are often used in different formats(i.e., abbreviated names). Co-referrences between those are important to be identified prior to IE tasks in order to avoid duplications or omissions of information.

Finally, the use of cue phrases and cue words for increasing the answer set worked well in our approach. Although the set is relatively small(we have something like 200 cue phrases defined) their identification need to be automated. To do this we have begun to work with a technique borrowed from the data engineering domain [Krulwich, 1995], which applies heuristics to identify 'semantically significant phrases'. The underlying principle is to observe visual effects often used by authors to emphasize important concepts in their documents. For example, boldfaced or italicised words, heavily repeated

phrases, compound noun phrases, list of items, etc. We have build a prototype tool which extracts a large set of potential cue phrases after applying a designated set of heuristics. The potential phrases will then be edited to construct the final set.

With this first version of *myPlanet* and the extensions we plan to make we are working towards the vision of the *Knowledge User* era where the user is the focal point in a setting with a plethora of knowledge-intensive systems aim to deliver intelligent services over the Web surrounding him. This metaphor, although in its infancy yet, is in contrast with the traditional view of knowledge-intensive systems being the focal point with users surrounding them acting as subscribers for knowledge services.

Acknowledgements

The research described in this paper is supported by the Interdisciplinary Research Collaboration(IRC) project: Advance Knowledge Technologies(AKT - http://www.aktors.org) funded by the UK government.

References

[Crow and Shadbolt, 1999] L. Crow and N. Shadbolt. Acquiring and Structuring Web Content with Knowledge Level Models. In R. Studer and D. Fensel, editors, *Proceedings of the 11th European Workshop on Knowledge Acquisition, Modelling and Management(EKAW'99), Dagstuhl, Germany*, pages 83–101. Springer Verlag, May 1999.

[Domingue and Motta, 2000] J. Domingue and E. Motta. Planet-Onto: From News Publishing to Integrated Knowledge Management Support. *IEEE Intelligent Systems*, 15(3):26–32, 2000.

[Domingue, 1998] J. Domingue. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. In *Proceedings of the 11th Knowledge Acquisition, Modelling and Management Workshop, KAW'98, Banff, Canada*, April 1998.

[Faatz *et al.*, 2000] A. Faatz, T. Kaamps, and R. Steinmetz. Background knowledge, indexing and matching interdependencies if document management and ontology maintenance. In *position paper in Proceedings of the ECAI2000 workshop on Ontology Learning(OL2000), Berlin, Germany*, August 2000.

[Guarino *et al.*, 1999] N. Guarino, C. Masolo, and G. Vetere. OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems*, 14(3):70–80, May 1999.

[Huhns and Stephens, 1999] M. Huhns and L. Stephens. Personal Ontologies. *IEEE Internet Computing*, 3(5):85–87, September 1999.

[Kalfoglou *et al.*, 2000] Y. Kalfoglou, T. Menzies, K-D. Althoff, and E. Motta. Meta-knowledge in systems design: panacea...or undelivered promise? *The Knowledge Engineering Review*, 15(4), 2000.

[Kalfoglou, 2000a] Y. Kalfoglou. *Deploying Ontologies in Software Design*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, June 2000.

[Kalfoglou, 2000b] Y. Kalfoglou. On the convergence of core technologies for knowledge management and organisational memories: ontologies and experience factories. In *Proceedings of the ECAI 2000 Workshop on Knowledge Management and Organisational Memories(W11-KMOM'00), Berlin, Germany*, August 2000.

[Krulwich and Burkley, 1997] B. Krulwich and C. Burkley. The InfoFinder Agent: Learning User Interests through Heuristic Phrase Extraction. *IEEE Intelligent Systems*, 12(5):22–27, 1997.

[Krulwich, 1995] B. Krulwich. Learning document category descriptions through the extraction of semantically significant phrases. In *Proceedings of the IJCAI'95 Workshop on Data Engineering for Inductive Learning*, 1995.

[McGuinness, 1998] L.D. McGuinness. Ontological Issues for Knowledge-Enhanced Search. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontology in Information Systems(FOIS'98), Trento, Italy*, pages 302–316. IOS Press, June 1998.

[Motta, 1999] E. Motta. *Reusable Components for Knowledge Models: Case Studies in Parametric Design Problem Solving*, volume 53 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1999. ISBN: 1-58603-003-5.

[O'Leary, 1998] D. O'Leary. Knowledge Management Systems: Converting and Connecting. *IEEE Intelligent Systems*, 13(3):30–33, June 1998.

[Proux and Chenevoy, 1997] D. Proux and Y. Chenevoy. Natural language processing for book storage: Automatic extraction of information from bibliographic notices. In *Proceedings of the Natural Language Processing Pacific Rim Symposium(NLPRS'97)*, pages 229–234, 1997.

[Riloff, 1996] E. Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *AI Journal*, (85):101–134, 1996.

[Roux *et al.*, 2000] C. Roux, D. Proux, F. Rechermann, and L. Julliard. An ontology enrichment method for a pragmatic information extraction system gathering data on genetic interactions. In *position paper in Proceedings of the ECAI2000 Workshop on Ontology Learning(OL2000), Berlin, Germany*. August 2000.

[Vargas-Vera *et al.*, 2001] M. Vargas-Vera, J. Domingue, Y. Kalfoglou, E. Motta, and S. Buckingham-Shum. Template-driven information extraction for populating ontologies. In *Proceedings of the IJCAI'01 Workshop on Ontology Learning, Seattle, WA, USA*, August 2001.

Combining and relating ontologies: an analysis of problems and solutions

Michel Klein
Vrije Universiteit Amsterdam
Michel.Klein@cs.vu.nl

Abstract

With the grown availability of large and specialized online ontologies, the questions about the combined use of independently developed ontologies have become even more important. Although there is already a lot of research done in this area, there are still many open questions. In this paper we try to classify the problems that may arise into a common framework. We then use that framework to examine several projects that aim at some ontology combination task, thus sketching the state of the art. We conclude with an overview of the different approaches and some recommandations for future research.

1 Introduction

In the last few years, there has been put a lot of effort in the development of techniques that aim at the "Semantic Web". This next step in the evolution of the World Wide Web, will enable computers to partly "understand" the information on the internet. A lot of those newly developed techniques requires and enables the specification of ontologies (Gruber, 1993) on the web. Consequently, there will emerge a lot of freely accessible domain specific ontologies. The reuse of these ontologies may be very attractive.

However, there are several problems when one tries to use independently developed ontologies together, or when existing ontologies are adapted for new purposes. Although there is already a lot of research done in this area, there are still many open questions. In this paper, we investigate the problems that may arise. We will distinguish several types of mismatches that can occur between different ontologies, we will look at practical problems and we will look at some of the consequences of changes to ontologies. Altogether, this will give us a framework that can be used to compare approaches that aim at solving the problems. We will use this to examine several techniques and tools that has the purpose to solve these problems or to support users in performing ontology combining tasks.

The paper is organized as follows. We will first clarify the terminology that is used in the field of ontology combining (section 2). In section 3, we will investigate all problems that arise when ontologies are combined or related, which results in a framework of relevant issues. Next, in section 4, we will use the framework to examine existing approaches for ontology combining. In section 5, we summarize the techniques and give an overview of the different approaches that are used. Finally, in section 6 we will conclude the paper and we will make some remarks based on our observations.

2 Terminology

Before we can analyse the problems that play a role, we need to clarify the terminology and define the terms we will use. We will have to make some decisions about our understanding of the terminology, because there is not always an agreement on the exact meaning of the terms. We have tried to be consistent as far as possible with definitions and descriptions found elsewhere.

Reuse of existing ontologies is often not possible without considerable effort (Uschold *et al.*, 1998). When one wants to reuse different ontologies together, those ontologies have to be *combined* in some way. This can be done by *integrating* (Pinto *et al.*, 1999) the ontologies, which means that they are merged into one new ontology, or the ontologies can be kept separate. In both cases, the ontologies have to be *aligned*, which means that they have to be brought into mutual agreement.

Ontology integration consist of (the iteration of) the following steps (McGuinness *et al.*, 2000):

1. find the places in the ontologies where they overlap;
2. relate concepts that are semantically close via equivalence and subsumption relations (aligning);
3. check the consistency, coherency and non-redundancy of the result.

The alignment of concepts between ontologies is especially difficult, because this requires understanding of the meaning of concepts. Aligning two ontologies implies changes to at least one of them. Changes to an ontology will result in a new *version* of an ontology.

If the ontologies are not represented in the same language, a *translation* is often required.

Throughout this paper, we will use the following terms consistently according to their specified meaning:

combining: Using two or more different ontologies for a task in which their mutual relation is relevant.

merging, integrating: Creating a new ontology from two or more existing ontologies with overlapping parts, which can be either virtual or physical.

aligning: Bring two or more ontologies into mutual agreement, making them consistent and coherent.

mapping: Relating similar (according to some metric) concepts or relations from different sources to each other by an equivalence relation. A mapping result in a virtual integration.

articulation: The points of linkage between two aligned ontologies, ie. the specification of the alignment.

translating: Changing the representation formalism of an ontology while preserving the semantics.

transforming: Changing the semantics of an ontology slightly (possibly also changing the representation) to make it suitable for purposes other than the original one.

version: The result of a change that may exist next to the original.

versioning: A method to keep the relation between newly created ontologies, the existing ones, and the data that conforms to them consistent.

3 Problems with ontology combination

The combined use of multiple ontologies is hindered by several problems. In this section, we will investigate and describe them.

The problems that underlies the difficulties in merging and aligning are the mismatches that may exist between separate ontologies. In the next subsection, we will discuss these mismatches. We will then look at the different type of problems involved with versioning and revisioning. Finally, we will discuss some practical problems that come up when one tries to combine ontologies.

Thus doing, we will build a framework with the different types of problems that can occur when relating ontologies. This framework can be used when we compare the existing approaches and tools.

3.1 Mismatches between ontologies

Mismatches between ontologies are the key type of problems that hinder the combined use of independently developed ontologies. We will now explore *how* ontologies may differ. In the literature, there are a lot of possible mismatches mentioned, which are not always easy comparable. To make them more comparable, we try to classify the different types of mismatches and relate them to each other.

As a first step, we will distinguish between two levels at which mismatches may appear. The first level is the **language** or meta-model level. This is the level of the language primitives that are used to specify an ontology. Mismatches at this level are mismatches between the *mechanims* to define classes, relations and so on. The second level is the **ontology** or model level, at which the actual ontology of a domain lives. A mismatch at this level is a difference in the way the domain is modelled. The distinction between these two levels of differences is made very often. Kitakami *et al.* (1996) and Visser *et al.* (1997) call these kinds of differences respectively *non-semantic* and *semantic* differences. Others make this distinction implicitly, by only concentrating on one of the two levels. For example, Wiederhold (1994) analyses domain differences (i.e., ontology level), while Grosso *et al.* (1998) and Bowers and Delcambre (2000) look at langauge level differences. In the following, we will avoid the use of the words "semantic differences" for ontology level differences, because we reserve those words for a more specific type of difference (which will be described below).

Below, we will give an overview and characterization of different types of mismatches that can appear at each of those two levels.

Language level mismatches

Mismatches at the language level occur when ontologies written in different ontology languages are combined. Chalupsky (2000) defines mismatches in *syntax* and *expressivity*. In total, we distinguish four types of mismatches that can occur, although they often coincide.

- **Syntax** Obviously, different ontology languages often use different syntaxes. For example, to define the class of chairs in RDF Schema (Brickley and Guha, 2000), one uses `<rdfs:Class ID="Chair">`. In LOOM, the expression `(defconcept Chair)` is used to define the same class. This difference is probably the most simple kind of mismatch. However, this mismatch often doesn't come alone, but is coupled with other differences at the language level. A typical example of a "syntax only" mismatch is an ontology language that has several syntactical representations. In this simple case, a rewrite mechanims is sufficient to solve those problems.

- **Logical representation** An slightly more complicated mismatches at this level is the difference in representation of logical notions. For example, in some languages it is possible to state explicitly that two classes are disjoint (e.g., `disjoint A B`), whereas it is necessary to use negation in subclass statements (e.g., `A subclass-of (NOT B), B subclass-of (NOT A)` in other languages. The point here is not whether something can be expressed — the statements are logically equivalent — but which langauge constructs should be used to express something. Also, notice that this mismatch is not about the representation of *concepts*, but about the representation of *logical notions*.

  This type of mismatch is still relatively easy solvable, e.g. by giving translation rules from one logical representation to another.

- **Semantics of primitives** A more subtle possible difference at the metamodel level is the semantics of language constructs. Despite the fact that sometimes the same name is used for a language construct in two languages,

the semantics may differ; e.g., there are several interpretations of `A equalTo B`.

Note that even when two ontologies seem to use the same syntax, the semantics can differ. For example, the OIL RDF Schema syntax (Broekstra *et al.*, 2001) interprets multiple `<rdfs:domain>` statements as the intersection of the arguments, whereas RDF Schema itself uses union semantics[1].

- **Language expressivity** The mismatch at the metamodel level with the most impact is the difference in expressivity between two languages. This difference implies that some langauges are able to express things that are not expressible in other languages. For example, some languages have constructs to express negation, others have not. Other typical differences in expressivity are the support of lists and sets, default values, etc.

  This type of mismatch has probably the most impact, and is mentioned by several others. The "fundamental differences" between knowledge models that are described in (Grosso *et al.*, 1998) are also very close to our interpretation.

Our list of differences at the language level can be seen as more or less compatible with the broad term "language heterogeneity" of Visser *et al.* (1997).

Ontology level mismatches

Mismatches at the ontology — or model — level happen when two or more ontologies that describe (partly) overlapping domains are combined. These mismatches may occur when the ontologies are written in the same language, as well as when they use different languages. Based on the literature and on our own observations, we can distinguish several types of mismatches at the model level.

Visser *et al.* (1997) make a very useful distinction between mismatches in the *conceptualization* and *explication* of ontologies. A conceptualization mismatch is a difference in the way a domain is interpreted (conceptualized), which results in different ontological concepts or different relations between those concepts. An explication mismatch, on the other hand, is a difference in the way the conceptualization is *specified*. This can manifest itself in mismatches in definitions, mismatches in terms and combinations of both. Visser *et al.* list all the combinations. Four of them are related to hymonym terms and synonym terms.

Wiederhold (1994) also mentions the problems with synonym terms (called *naming differences*) and homonym terms (*subjective meaning*). Besides that, he describes possible differences in the *scope of concepts*, which is an example of a conceptual mismatch. Finally, he mentions *value encoding* differences, for example, differences in the currency of prices.

Chalupsky (2000) list four types of mismatches in ontologies. One of them, *inference system bias* is in our opinion not a real mismatch, but a reason for modeling style differences. The other three mismatches, *modeling conventions*, *coverage and granularity* and *paradigms* can be categorized as instances of the two main mismatch types of Visser *et al.*. We will describe them in a slightly altered way below.

We will now relate the different types of mismatches that are distinguished by the authors cited above. Thus, we will continue the build of our framework.

The first two mismatches at the model level that we distinguish are instances of the **conceptualization mismatches** of Visser *et al.*. This are semantic differences, i.e., not only the specification, but also the conceptualization of the domain (see the definition of Gruber, 1993) is different in the ontologies that are involved.

- **Scope** Two classes seem to represent the same concept, but do not have exactly the same instances, although these intersect. The standard example is the class "employee": several administrations use slightly different concepts of employee, as mentioned by Wiederhold (1994). In (Visser *et al.*, 1997), this is called a *class mismatch* and is worked out further into detailed descriptions at class- or relation-level.
- **Model coverage and granularity** This is a mismatch in the part of the domain that is covered by the ontology, or the level of detail to which that domain is modelled. Chalupsky (2000) gives the example of an ontology about cars: one ontology might model cars but not trucks. Another one might represent trucks but only classify them into a few categories, while a third one might make very fine-grained distrinctions between types of trucks based on their general physical structure, weight, purpose, etc.

Conceptualization differences as described above can not be solved automatically, but require knowledge and decisions of a domain expert. In the second case, the mismatch is often not a problem, but a motive to use different ontologies together. In that case, the remaining problem is to align the overlapping parts of the ontology.

The other ontology-level mismatches can be categorized as **explication mismatches**, in the terminology of Visser *et al.*. The first two of them result from explicit choices of the modeler about the **style of modeling**:

- **Paradigm** Different paradigms can be used to represent concepts such as time, action, plans, causality, propositional attitudes, etc. For example, one model might use temporal representations based on interval logic while another might use a representation based on point (Chalupsky, 2000). The use of different "top-level" ontology is also an example of this kind of mismatch.
- **Concept description** This type of differences are called *modeling conventions* in (Chalupsky, 2000). Several choices can be made for the modeling of concepts in the ontology. For example, a distinctions between two classes can be modeled using a qualifying attribute or by introducing a separate class. These choices are sometimes influenced by the intended inference system. Another choice in concept descriptions is the way in which is-a hierarchy is build; distinctions between features can be made higher or lower in the hierarchy. For example, consider the place where the distinction between

[1] Although this will probably change in the next revision of RDF Schema, according to a discussion on the RDF-interest mailinglist.

scientific and non-scientific publications is made: a dissertation can be modeled as `dissertation < book < scientific publication < publication`, or as `dissertation < scientific book < book < publication`, or even as subclass of both `book` and `scientific publication`.

Further, the next two types of differences can be classified as **terminological mismatches**.

- **Synonym terms** Concepts are represented by different names. A trivial example is the use of the term "car" in one ontology and the term "automobile" in another ontology. This type of problem is called *term mismatch (T or TD)* in (Visser *et al.*, 1997).

  A special type of this problem is the case that the natural language in which ontologies are described differ.

  Although the technical solution for this type of problems seems relatively simple (the use of thesauri), the integration of ontologies with synonyms or different languages requires usually a lot of human effort and comes with several semantic problems. Especially, one must be careful not to overlook a scope difference (see above).

- **Homonym terms** The meaning of a term is different in an other context. For example, the term "conductor" has a different meaning in a music domain than in an electric engineering domain. Visser *et al.* (1997) calls this a *concept mismatch (C or CD)*.

  This inconsistency is much harder to handle; (human) knowledge is required to solve this ambiguity.

Finally, there is a one trivial type of difference left.

- **Encoding** Values in the ontologies may be encoded in different formats. For example, a date may be represented as "dd/mm/yyyy" or as "mm-dd-yy", distance may be described in miles or kilometers, etc. There are many mismatches of this type, but these are all very easy to solve. In most cases, a transformation step or wrapper is sufficient to eliminate all those differences.

3.2 Ontology versioning

The problems listed above are mismatches between ontologies. Most projects and approaches focus on solving these mismatches. However, mismatches are not the only problems that have to be solved when one want to use several ontologies together for one task.

As changes to ontologies are inevitable in an open domain, it becomes very important to keep track of the changes and of its impact on the dependencies of that ontology. It is often not practically possible to synchronize the changes of an ontology with the revisions to the applications and datasources that use them. Therefore, a versioning method is needed to handle revisions of ontologies and the impact on existing sources. In some sense, the versioning problem can also be regarded as a derivation of ontology combination; it results from changes (possibly required by combination tasks) to individual ontologies.

Ontology versioning covers several aspects. Although the problem is introduced by subsequent changes to one specific ontology, the most important problems are caused by the dependencies on that ontology. Therefore, it is useful to distinguish the aspects of ontology versioning. A versioning scheme should take care of the following aspects:

1. the relation between succeeding revisions of one ontology;
2. the relation between the ontology and its dependencies:
   - instance data that conforms to the ontology;
   - other ontologies that are built from, or import the ontology;
   - applications that use the ontology.

The central question that a versioning scheme answers is: how to reuse existing ontologies in new situations, without invalidating the existing ones. A versioning scheme provides ways to disambiguate the interpretation of concepts for users of the ontology revisions, and it makes the compatibility of the revisions explicit. Consequently, we can impose the following requirements on a versioning scheme, in increasing level of difficulty:

- for every use of a concept or a relation, a versioning framework should provide an unambigious reference to the intended definition (**identification**);
- a versioning framework should make the relation of one version of a concept or relation to other versions of that construct explicit (**change tracking**);
- a versioning framework should — as far as possible — automatically perform conversions from one version to another, to enable transparant access (**transparant translating**).

We will examine the current approaches with respect to those requirements.

3.3 Practical problems

Besides the technical problems that we discussed in the previous sections, there are also practical problems that hinder the easy use of combined ontologies. Aligning and merging ontologies, the central aspect of ontology combining, is a complicated process and requires serious effort of the ontology designers. Until now, this task is mostly done by hand (Noy and Musen, 2000), which makes it difficult to overlook in two aspects:

- it is difficult to find the terms that need to be aligned;
- the consequences of a specific mapping (unforeseen implications) are difficult to see.

Because it is unrealistic to hope that merging or alignment at the semantic level could be performed completely automatically, we should take these practical aspects into consideration.

Another practical problem is that repeatability of merges. Most often, the sources that are used for the merging, continue to evolve. The alignments that are created for the merging, should be as much reusable as possible for the merging of the revised ontologies. This issue is very important in the context of ontology maintenance. The repeatability could for example be achieved by an executable

specification of the alignment.

Summarizing the previous sections, we can construct the framework of issues that is depicted in Figure 1.

4 Current approaches and techniques

In this section, we will use the framework to examine several tools and techniques that are aimed at ontology combining. We start at the top of the framework, looking at techniques for solving language mismatches. Then, we will look at techniques for solving ontology level mismatches and user support. Of course, it is not possible to make a strict distinction between the type of problems that a technique solves, because some tools or techniques provide support for several types of problems. The place where we mention them does therefore not imply a classification, but serves as a guideline only.

4.1 Solving language mismatches

There are several approaches for solving the problem of integrating ontologies that are written in different knowledge representation languages. Some of them are just techniques, others also provide some kind of automated support.

Superimposed metamodel

Bowers and Delcambre (2000) describes an approach to transforming information from one representation to another. Their focus is on model-based information where the information representation scheme provides structural modeling constructs (analogous to a data model in a database). For example, the XML model includes elements, attributes, and permits elements to be nested. Similarly, RDF models information through resources and properties. The goal of their work is to enable the user to apply tools of interest to the information at hand. Their approach is to represent information for a wide variety of model-based applications in a uniform way, and to provide a mapping formalism that can easily transform information from one representation to another.

To achieve this, the ontology languages (i.e., the specific constructs in the language that are used to describe an ontology) are each represented in a meta-model, a so called ”superimposed” model. These superimposed models are represented as RDF triples. Mappings are then specified using production rules. The rules are defined over triples of the RDF representation for superimposed information. Since a triple is a simple predicate, mapping rules are specified using a logic-based language such as Prolog, which allows to both specify and implement the mappings. There is no requirement that the mappings between superimposed layers should be complete, since only part of a model or schema may be needed while using a specific tool.

If superimposed information from a source language is being mapped to the target language, is is possible to convert *data* from the source layer into data that conforms to the target layer. Although the focus is on conversion, it is also possible to perform integration between superimposed layers. Integration goes a step further by combining the source and target data. The mapping rules can be used to provide integration at both the schema and instance levels.

The “superimposed model approach” provides mechanisms to solve language level mismatches of syntax, representation and semantics. The mappings between the language constructs have to be specified manually. The semantic resolution of mismatches at the ontology level is not covered by this approach.

Layered approach to interoperability

Melnik and Decker (2000) introduce some initial ideas targeted at facilitating data interoperation using a layered approach. Their approach resembles the layering principles used in internetworking. To harness the complexity of data interoperation, Melnik and Decker suggest viewing Web-enabled information models as a series of layers: the syntax layer, the object layer, and the semantic layer. The semantic layer, or knowledge representation layer, deals with conceptual modeling and knowledge engineering tasks. The basic function of the object layer, or frame layer, is to provide applications with an object-oriented view of their domain. The syntax layer is responsible for ”dumbing down” object-oriented information into document instances and byte streams. Each layers has a number of sublayers, which corresponds to a specific data modeling feature (e.g., aggregation or reification) that can be logically implemented in different ways.

It seems that a clean separation between different layers may ease the achievement of interoperability. The first two layers that the authors distinguish map nicely onto the first two types of mismatches that we described in Section 3.1. However, all other types of mismatches that we distinguish are comprised in the “semantic layer”. Therefore, the layering as described in its inital version only solves some of the language level mismatches.

As the authors also have noticed, considering data models in a layered fashion is a contemporary approach. For example, OIL is presented as an extension to RDF Schema (Broekstra *et al.*, 2001), and Euzenat (2001) investigates the characteristics of interoperability of knowledge representations at various levels.

OKBC

The Open Knowledge Base Connectivity (Chaudhri *et al.*, 1998) is a generic interface to knowledge representation systems (KRS). An “application programming interface” (API), specifies the operations that can be used to access a system by an application program. When specifying this API for knowledge representation systems, some assumptions are made about the representation used by that KRS. These assumptions are made explicit in the OKBC knowledge model.

A specific knowledge representation language — or ontology language — can be bound to OKBC by defining a mapping from OKBC knowledge model to the specific language. The users of the ontology are then isolated from the peculiarities of specific language and can use the OKBC model. The interoperability achieved by using OKBC is at the level of the OKBC knowledge model.

OKBC thus can solve some mismatches at the language level. However, semantic differences that are beyond representation can not be solved by providing mappings to the

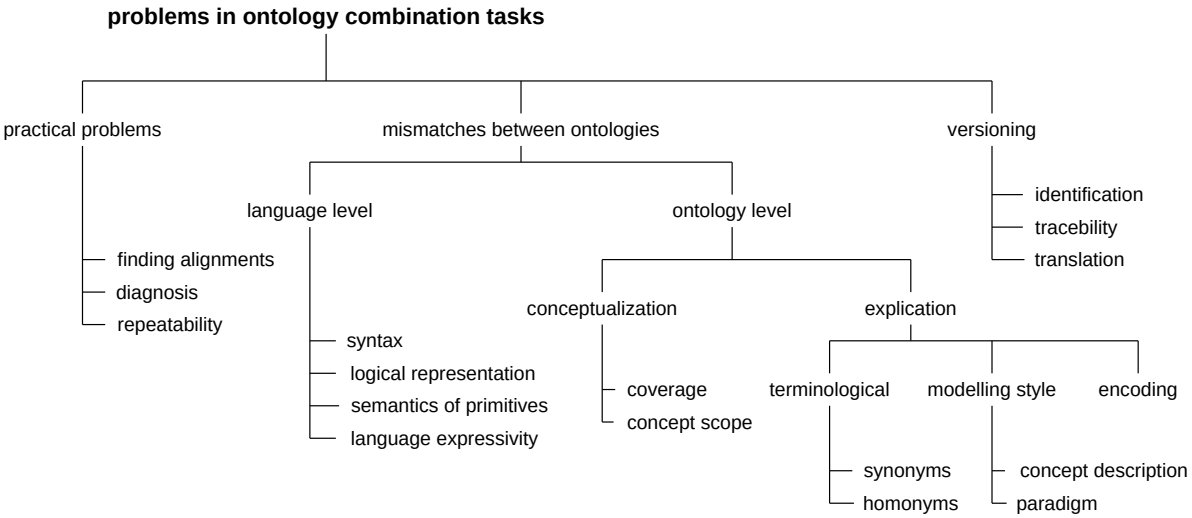


Figure 1: The resulting framework of issues that are involved in ontology combining

OKBC knowledge model. More general, when using a mapping to a common knowledge model, the notions that requires a higher level of expressivity than that model provides, will be lost.

OntoMorph

OntoMorph (Chalupsky, 2000) is a transformation system for symbolic knowledge. It facilitates ontology merging and the rapid generation of knowledge base translators. It combines two mechanisms to describe knowledge base transformations: (1) *syntactic rewriting* via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching, and (2) *semantic rewriting* which modulates sytactic rewriting via (partial) semantic models and logical inference via an integrated KR system. The integration of these mechanisms allows transformations to be based on any mixture of syntactic and semantic criteria. The OntoMorph architecture facilitates incremental development and scripted replay of transformations, which is particulary important during merging operations.

OntoMorph focuses at transformations to individual ontologies that are needed to align two or more ontologies. This is small but important step in the process of merging ontologies. In fact, step number 2 of the ontology merging process (see Section 2) is split into three:

2a. design transformations to bring the sources into mutual agreement;

2b. editing or *morphing* the sources to carry out the transformations;

2c. taking the union of the morphed sources;

OntoMorph facilitates step 2b, by transforming the ontologies into a common format with common names, common syntax, uniform modeling assumptions, etc.

Step 2a, the design of the transformations, involves understanding of the meaning of the representation, and is therefore a less automatable task. Additionally, this step often involves human negotiation to reconcile competing views on how a particular modeling problem should be solved.

OntoMorph is able to solve several problems at the language level of ontology mismatches framework. Of course, a difference in expressivity between two languages is not solvable, but some implies loss of knowledge. Solutions for ontology level problems can also be formulated in OntoMorph. Because OntoMorph requires a clear and executable specification of the transformation, the process can be repeated with modified versions of the original ontologies.

4.2 Ontology level integration and user support

In the previous section, we saw that OntoMorph provide mechanisms and support for some model level integration, too. We will now look at a transformation system, that allows the specification and execution of transformation of individual ontologies. We will then discuss two tools that assist the user in the complicated task of performing a merge.

Algebra for Scalable Knowledge Composition

The Scalable Knowledge Composition (SKC)[2] project developed an *algebra* for ontology composition. This algebra is used in the ONION system (ONtology compositION), described in (Mitra *et al.*, 2000). The current work in the SKC project is not solely in the area of ontology combination, but in the broader field of integrating heterogenous datasources.

The algebra operates on ontologes that are represented by nodes and arcs (terms and relationships) in a directed labelled graph. Each algebraic operator takes as input a graph of semistructured data and transforms it to another graph. This guarantees that the algebra is composable. The algebra operations are themselves knowledge driven, using articulation rules. The rules can be both logical rules (e.g., semantic

[2]See `http://www-db.stanford.edu/SKC/`.

implication between terms across ontologies) and functional rules (e.g., dealing with conversion between terms across ontologies). The composition rules are partly suggested by expert and lexical knowledge.

Intersection is the most crucial operation since it identifies the terms where linkage occurs among the domains, which is called "the articulation". An *articulation ontology* contains the terms from the source ontologies that are related and their relation, and can be seen as a specification of an alignment. This separate specification facilitates repeated executions of the composition.

When we relate this to our framework, we see that the algebra allows the specification of solutions to solve several conceptual and terminological mismatches. Via the functional rules, term synonyms and encoding problems can be eliminated. The logical articulation rules provides a mean to solve mismatches in scope, coverage and homonym terms. By using expert and lexical knowledge to suggest articulations, the system also meets the practical problems of finding alignments.

One of the main advantages of using an algebra for combination, is the reusability. The unified ontology is not an physical entity, but an term to denote the result of applying the articulation rules. This approach ensures minimal coupling between the sources, so that the sources can be developed and maintained independently.

Chimaera

Chimaera (McGuinness *et al.*, 2000) is an ontology merging and diagnosis tool developed by the Stanford University Knowledge Systems Laboratory (KSL). Its initial design goal was to provide substantial assistance with the task of merging KBs produced by multiple authors in multiple settings. Later, it took on another goal of supporting testing and diagnosing ontologies as well. Finally, inherent in the goals of supporting merging and diagnosis are requirements for ontology browsing and editing. It is mainly targeted at lightweight ontologies.

The two major tasks in merging ontologies that Chimaera support are (1) coalesce two semantically identical terms from different ontologies so that they are referred to by the same name in the resulting ontology, and (2) identify terms that should be related by subsumption, disjointness, or instance relationships and provide support for introducing those relationships. There are many auxiliary tasks inherent in these tasks, such as identifying the locations for editing, performing the edits, identifying when two terms could be identical if they had small modifications such as a further specialization on a value-type constraint, etc.

Chimaera generates name resolution lists that help the user in the merging task by suggesting terms each of which is from a different ontology that are candidates to be merged or to have taxonomic relationships not yet included in the merged ontology. It also generates a taxonomy resolution list where it suggests taxonomy areas that are candidates for reorganization. It uses a number of heuristic strategies for finding such edit points.

Finally, Chimaere also has diagnostic support for verifying, validating, and critiquing ontologies. Those functions only include domain independent tests that showed value in previous experiments.

We see that Chimaera can be used to solve mismatches at the terminological level. It is also able to find some similar concepts that have a different description at the model level. Further, Chimaera seems to do a great job in helping the user to find possible edit point. The diagnostic functions are difficult to evaluate, because their description is very brief.

PROMPT

PROMPT (formerly known as SMART) is an interactive ontology-merging tool (Noy and Musen, 2000). It guides the user through the merging process making suggestions, determining conflicts, and proposing conflict-resolution strategies. PROMPT starts with the linguistic-similarity matches of frame names for the initial comparison, but concentrates on finding clues based on the structure of the ontology and users actions. After the user selects an operation to perform, PROMPT determines the conflicts in the merged ontology that the operation have caused and proposes possible solutions to the conflict. It then considers the structure of the ontology around the arguments to the latest operations — relations among the arguments and other concepts in the ontology — and proposes other operations that the user should perform.

In the PROMPT project, a set of knowledge-base operations for ontology merging or alignment is identified. For each operation in this set is defined (1) the changes that PROMPT performs automatically, (2) the new suggestions that PROMPT presents to the user, and (3) the conflicts that the operation may introduce and that the user needs to resolve. When the user invokes an operation, PROMPT creates members of these three sets based on the arguments to the specific invocation of the operation.

The conflicts that may appear in the merged ontology as the result of these operations are: name conflicts, dangling references, redundancy in the class-hierarchy and inconsistencies. PROMPT not only points to the places where changes should be made, but also presents a list of actions to the user.

Summarizing, PROMPT gives iterative suggestions for concept merges and changes, based on linguistic and structural knowledge, and it points the user to possible effects of these changes.

Common top level model

A different approach for enabling model level interoperability, is the use of a common top level ontology. One of the project that takes this approach is ABC (Brickley *et al.*, 1999), a common conceptual model to facilitate interoperability among application metadata vocabularies.

ABC aims at the interoperability of multiple metadata packages that may be associated with and across resources. These packages are by nature not semantically distinct, but overlap and relate to each other in numerous ways. It exploits the fact that many entities and relationships - for example, people, places, creations, organizations, events, certain relationships and the like - are so frequently encountered that they do not fall clearly into the domain of any particular metadata vocabulary but apply across all of them. ABC is an attempt to:

- formally define these underlying common entities and relationships;
- describe them (and their inter-relationships) in a s/usr/bin/smbclient -M pelikaanimple logical model;
- provide the framework for extending these common semantics to domain and application-specific metadata vocabularies.

A comparable approach - although more general - is the IEEE Standard Upper Ontology (IEEE SUO Working Group). This standard will specify the semantics of a general-purpose upper level ontology. This will be limited to the upper level, which provides definition for general-purpose terms and provides a structure for compliant lower level domain ontologies. It is estimated to contain between 1000 and 2500 terms plus roughly ten definitional statements for each term. Is intended to provide the foundation for ontologies of much larger size and more specific scope.

These approaches can solve some interoperability, but requires a manual mapping of the ontologies to the common ontology.

4.3 Versioning

We only found one technique that provides support for the ontology versioning problems. Of course, there is a lot of experience with these kind of problems in the area of software engineering and databases, but it is not yet clear whether this can directly applied to web-based ontologies. This should be investigated further.

SHOE ontology integration

SHOE (Heflin and Hendler, 2000) is an is an HTML-based ontology language. The language includes techniques for combining and integrating different ontologies. SHOE provides a rule mechanism to align ontologies. Common items between ontologies can be mapped by inference rules. First, terminological differences can be mapped using simple if-and-only-if rules. Second, scope differences require mapping a category to the most specific category in the other domain that subsumes it. Third, some encoding differences can be handled by mapping individual values. Not all encodings can be mapped in SHOE, for example arithmetic functions would be needed to map meters to feet.

To solve versioning problems, the SHOE project gives version numbers to ontologies and suggest three ways to incorporate the results of an ontology integration effort. These revising schemes allows for the different effects of revisions on the compatibility.

- In the first approach, a new mapping ontology that extends all the existing ones is created; users of the integrated ontology should explicitly conform to the newly created ontology.
- Second, each ontology that is involved in the integration could be revised with the mutual relations to the other ontologies.
- Third, it is possible to create a new intersection ontology, that will be extended by the already existing ontologies. Any source can commit to the ontology of its choice, thus saying that it agrees with any conclusions that logically follow from its statements and the rules in the ontology. Agents are free to pick which ontologies they use to interpret a source, and depending on the differences between these two ontologies they may get the intended meaning or an alternate one.

The SHOE versioning facilities provides both identification of the revisions and an explicit specification of its relation to other versions.

5 Overview of approaches

We will now give an overview of the approaches that are used in the projects and techniques mentioned above and also list some papers that describe a similar approach.

Additionally, Table 1 relates the the tools and approaches that we have discussed to the framework. The table should read as follows: an ‘A’ means ”tool or technique solves this without user interaction (automatically)”, ‘U’ means ”tool or technique suggests solutions to the user, and ‘M’ means ”tool or technique provides a mechanism for specifying the solution to this problem”.

- We discovered four different approaches that aims at enabling **interoperability** between different ontologies at the *language level*.
  - aligning the metamodel: the constructs in the language are formally specified in a general model (Bowers and Delcambre, 2000), (MOF);
  - layered interoperability: aspects of the language are split up in clearly defined layers, as a result of what interoperability can be solved layer by layer (Melnik and Decker, 2000);
  - transformation rules: the relation between two specific constructs in different ontology languages is described with a rule that specifies the transformation from the one to the other (OntoMorph);
  - mapping onto a common knowledge model: the constructs of an ontology language are mapped onto a common knowledge model (OKBC).

  Note that the third approach can be used to implement the fourth.
- We want to recall that the alignment of concepts is a task that requires understanding of the meaning of concepts, and cannot be fully automated. Consequently, at the *model level*, we only found tools that **suggest alignments and mappings** based on heuristics matching algorithms and provide means to specify these mappings. Such tools support the user in finding the concepts in the separate ontologies that might be candidates for merging. Some tools go a little bit further by even suggesting the actions that should be performed. Roughly spoken, there are two types of heuristics:
  - linguistic based matches: terms with the same word-stem, nearby terms in WordNet, or even simpler heuristics, like omitting hyphens and capitalizing all terms, etc. Examples can be found in (Hovy, 1998; Knight and Luk, 1994);

Table 1: Table of problems and approaches for combined use of ontologies

Issues		SKC	Chim.	PROMPT	SHOE	OntoM.	Metamodel	OKBC	Layering
Language level mismatches	Syntax					M	M	M	M
	Representation					M	M	M	M
	Semantics					M	M		M
	Expressivity								
Ontology level mismatches	Paradigm					M			
	Concept description					M			
	Coverage of model								
	Scope of concepts	M	U	U	M	M			
	Synonyms	M	U	U	M	M			
	Homonyms	M				U			
	Encoding	M			M	M			
Practical problems	Finding alignments	U	U	U					
	Diagnosis of results		A	A		A			
	Repeatability	A		A		A			
Ontology versioning	Identification				M				
	Change tracking				M				
	Translation								

  - structural and model similarity: see for example the techniques described in Chimaera and Weinstein and Birmingham (1999).

- A slightly different approach for interoperability at the model is the use of a **common top level ontology**. This approach is only useful if there is a willingness to conform to a common standard.
- There are also different approaches for **diagnosing** or **checking** the results of the alignments. We have seen two types of checks:
  - domain independent verification and validation checks: name conflicts, dangling references, etc. (Chimaera, and others);
  - validation that requires some kind of reasoning: redundancy in the class hierarchy, value restrictions that violate class inheritance, etc. (OntoMorph, PROMPT).
- Several tools support an **executable specification** of the mappings and transformations (SKC, OntoMorph, PROMPT). This allows re-merging of revised ontologies. In this way, the intellectual effort that is invested in finding and formulation the alignments is preserved.
- Finally, most techniques and tools do not deal with **versioning**. Only SHOE elaborates on schemes that enables the combined use of different ontologies. They mention three ways to integrate separate (revisions of) ontologies without invalidating the existing ones.

6 Conclusion and remarks

In this paper, we analyzed the problems that hinder the combined use of ontologies. These problems are of several kinds and may occur at several levels. The analyse of the problems yielded a framework that is used to examine what solutions are provided by current tools and techniques. This examination is still very general, and will be worked out further in the future.

We have seen that there are several approaches that provide reasonable support for language level interoperability. Mismatches in expressiveness between languages are not solvable, and consequently, none of the approaches takes this into account.

The most difficult problems are those of conceptual integration. There are a lot of techniques and heuristics for suggesting alignments. We think that semantic mapping at the model level will remain a task that requires a certain level of human intervention.

Finally, in an open environment such as the Web, versioning methods will be very important. We have seen that this aspect is underdeveloped in most approaches. We think that more comprehensive schemes for interoperability of ontologies are required.

Acknowledgements

We would like to thank Dieter Fensel, Mike Uschold for helpful comments and remarks on previous versions of this paper.

References

Shawn Bowers and Lois Delcambre. Representing and transforming model-based information. In *First Workshop on the Semantic Web at the Fourth European Conference on Digital Libraries*, Lisbon, Portugal, September 18–20, 2000.

D. Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Wide Web Consortium, March 2000.

Dan Brickley, Jane Hunter, and Carl Lagoze. ABC: A logical model for metadata interoperability, October 1999. Harmony discussion note, see `http://www.ilrt.bris.ac.uk/discovery/harmony/docs/abc/abc_draft.html`.

Jeen Broekstra, Michel Klein, Stefan Decker, Dieter Fensel, Frank van Harmelen, and Ian Horrocks. Enabling knowledge representation on the web by extending RDF schema. In *Proceedings of the 10th World Wide Web conference*, Hong Kong, China, May 1–5, 2001.

Hans Chalupsky. OntoMorph: A translation system for symbolic logic. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 471–482, San Francisco, CA, 2000. Morgan Kaufmann.

Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 600–607, Menlo Park, July 26–30 1998. AAAI Press.

Jérôme Euzenat. Towards a principled approach to semantic interoperability. In Asuncion Gomez-Perez, Michael Gruninger, Heiner Stuckenschmidt, and Michael Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, August 4–5, 2001.

Norman Foo. Ontology revision. In Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa, editors, *Proceedings of the 3rd International Conference on Conceptual Structures (ICCS'95): Applications, Implementation and Theory*, volume 954 of *LNAI*, pages 16–31, Berlin, GER, August 1995. Springer.

William E. Grosso, John H. Gennari, Ray W. Fergerson, and Mark A. Musen. When knowledge models collide (how it happens and what to do). In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW '98)*, Banff, Canada, April 18–23 1998.

T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.

Jeff Heflin and James Hendler. Dynamic ontologies on the web. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449. AAAI/MIT Press, Menlo Park, CA, 2000.

E. H. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain, May 28–30 1998.

IEEE SUO Working Group. Standard upper ontology. See `http://suo.ieee.org/`.

H. Kitakami, Y. Mori, and M. Arikawa. An intelligent system for integrating autonomous nomenclature databases in semantic heterogeneity. In *Database and Expert System Applications, DEXA'96*, number 1134 in Lecture Notes in Computer Science, pages 187–196, Zürich, Switzerland, 1996.

Kevin Knight and Steve K. Luk. Building a large-scale knowledge base for machine translation. In *Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1*, pages 773–778, Menlo Park, CA, USA, July 31–August 4 1994. AAAI Press.

Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 483–493, San Francisco, 2000. Morgan Kaufmann.

Sergey Melnik and Stefan Decker. A layered approach to information modeling and interoperability on the web. In *Electronic proceedings of the ECDL 2000 Workshop on the Semantic Web*, Lisbon, Portugal, September 21 2000.

Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings of Conference on Extending Database Technology, (EDBT 2000)*, Konstanz, Germany, March 2000. Also, Stanford University Technical Note, CSL-TN-99-411, August, 1999.

Natalya Fridman Noy and Mark A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, 2000. AAAI/MIT Press.

D. E. Oliver, Y. Shahar, M. A. Musen, and E. H. Shortliffe. Representation of change in controlled medical terminologies. *Artificial Intelligence in Medicine*, 15(1):53–76, 1999.

H. Sofia Pinto, Asunción Gómez-Pérez, and João P. Martins. Some issues on ontology integration. In *Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99*, Stockholm, Sweden, August 1999.

Mike Uschold, Mike Healy, Keith Williamson, Peter Clark, and Steven Woods. Ontology reuse and application. In N. Guarino, editor, *Formal Ontology in Information Systems (FOIS'98)*, Treno, Italy, June 6-8, 1998. IOS Press, Amsterdam.

Pepijn R. S. Visser, Dean M. Jones, T. J. M. Bench-Capon, and M. J. R. Shave. An analysis of ontological mismatches: Heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA, 1997.

Peter C. Weinstein and William P. Birmingham. Comparing concepts in differentiated ontologies. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW '99)*, Banff, Canada, October 16–21, 1999.

Gio Wiederhold. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, pages 56–61, U.S. Naval Postgraduate School, Monterey CA, September 1994.

Anchor-PROMPT: Using Non-Local Context for Semantic Matching

Natalya F. Noy and Mark A. Musen

Stanford Medical Informatics, Stanford University, Stanford, CA 94305-5479
{noy, musen}@smi.stanford.edu

Abstract

Researchers in the ontology-design field have developed the content for ontologies in many domain areas. Recently, ontologies have become increasingly common on the World-Wide Web where they provide semantics for annotations in Web pages. This distributed nature of ontology development has led to a large number of ontologies covering overlapping domains, which researchers now need to merge or align to one another. The processes of ontology alignment and merging are usually handled manually and often constitute a large and tedious portion of the sharing process. We have developed and implemented Anchor-PROMPT—an algorithm that finds semantically similar terms automatically. Anchor-PROMPT takes as input a set of anchors—pairs of related terms defined by the user or automatically identified by lexical matching. Anchor-PROMPT treats an ontology as a graph with classes as nodes and slots as links. The algorithm analyzes the paths in the subgraph limited by the anchors and determines which classes frequently appear in similar positions on similar paths. These classes are likely to represent semantically similar concepts. Our experiments show that when we use Anchor-PROMPT with ontologies developed independently by different groups of researchers, 75% of its results are correct.

1 Ontology Merging and Anchor-PROMPT

Researchers have pursued development of ontologies—explicit formal specifications of domains of discourse—on the premise that ontologies facilitate knowledge sharing and reuse (Musen 1992; Gruber 1993). Today, ontology development is moving from academic knowledge-representation projects to the world of e-commerce. Companies use ontologies to share information and to guide customers through their Web sites. The ontologies on the World-Wide Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (such as on Amazon.com). The WWW Consortium is developing the Resource Description Framework (Brickley and Guha 1999), a language for encoding semantic information on Web pages in machine-readable form. Such encoding makes it possible for electronic agents searching for information to share the common understanding of the semantics of the data represented on the Web. Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED (Price and Spackman 2000) and the semantic network of the Unified Medical Language System (Humphreys and Lindberg 1993).

With this widespread distributed use of ontologies, different parties inevitably develop ontologies with overlapping content. For example, both Yahoo! and the DMOZ Open Directory (Netscape 1999) categorize information available on the Web. The two resulting directories are similar, but also have many differences.

Currently, there are extremely few theories or methods that facilitate or automate the process of reconciling disparate ontologies. Ontology management today is mostly a manual process. A domain expert who wants to determine a correlation between two ontologies must find all the concepts in the two source ontologies that are similar to one another, determine what the similarities are, and either change the source ontologies to remove the overlaps or record a mapping between the sources for future reference. This process is both labor-intensive and error-prone.

The semi-automated approaches to ontology merging that do exist today (Section 2) such as PROMPT and Chimaera analyze only **local context** in ontology structure: given two similar classes, the algorithms consider classes and slots that are *directly* related to the classes in question. The algorithm that we present here, Anchor-PROMPT, uses a set of heuristics to analyze **non-local context**.

The goal of Anchor-PROMPT is not to provide a complete solution to automated ontology merging but rather to augment existing methods, like PROMPT and Chimaera, by determining additional possible points of similarity between ontologies.

Anchor-PROMPT takes as input a set of pairs of related terms—**anchors**—from the source ontologies. Either the user identifies the anchors manually or the system generates them automatically. From this set of previously identified anchors, Anchor-PROMPT produces a set of new pairs of semantically close terms. To do that, Anchor-PROMPT traverses the paths between the anchors in the corresponding ontologies. A path follows the links between classes defined by the hierarchical relations or by slots and their domains and ranges. Anchor-PROMPT then compares the terms along these paths to find similar terms.

For example, suppose we identify two pairs of anchors: classes A and B and classes H and G (Figure 1). That is, a class A from one ontology is similar to a class B in the other ontology; and a class H from the first ontology is similar to a class G from the second one. Figure 1 shows

one path from A to H in the first ontology and one path from B to G in the second ontology. We traverse the two paths in parallel, incrementing the similarity score between each two classes that we reach in the same step. For example, after traversing the paths in Figure 1, we increment the similarity score between the classes C and D and between the classes E and F. We repeat the process for all the existing paths that originate and terminate in the anchor points, cumulatively aggregating the similarity score.

The central observation behind Anchor-PROMPT is that if two pairs of terms from the source ontologies are similar and there are paths connecting the terms, then the elements in those paths are often similar as well. Therefore, from a small set of previously identified related terms, Anchor-PROMPT is able to suggest a large number of terms that are likely to be semantically similar as well.

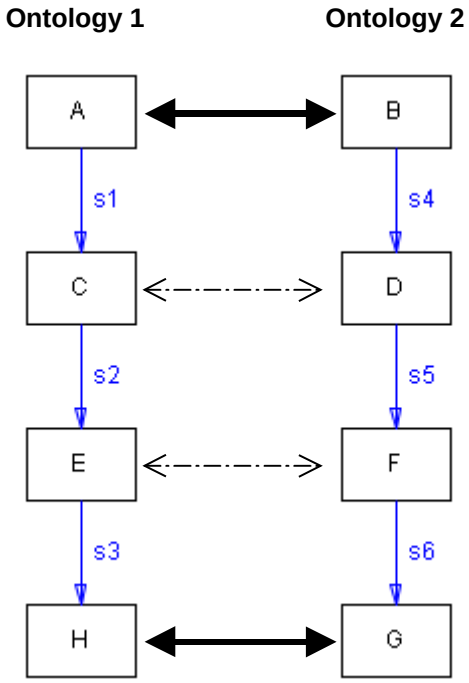


Figure 1. Traversing the paths between anchors. The rectangles represent classes and labeled edges represent slots that relate classes to one another. The left part of the figure represents classes and slots from one ontology; the right part represents classes and slots from the other. Solid arrows connect pairs of anchors; dashed arrows connect pairs of related terms.

2 Related Work

To date, researchers working on tools for ontology merging have expended their greatest effort finding mostly lexical matches among concepts in the source ontologies. Such systems usually rely on dictionaries to determine synonyms, evaluate common substrings, consider concepts whose documentation shares many uncommon words, and so on (Chapulsky et al. 1997; Wiederhold and Jannink 1999). These approaches, however, do not take into account the internal structure of concept representation, the structure of an ontology itself, or the steps users take during merging.

Researchers in the database community have addressed the problem of finding semantically similar terms in automating the process of matching database schemas. A number of schema-matching approaches use not only syntactic information (the similarity of the term names) but also the types of relations among terms. For example, the Artemis system (Castano and De Antonellis 1999) uses thesauri to determine lexical affinity between terms and combines uses domain types of schema elements with user input to determine structural affinity. The TransScm system (Milo and Zohar 1998) traverses the graph representation of two schemas performing a node-by-node comparison. However, the TransScm approach works well only if the input schemas have an extremely high degree of similarity.

The Chimaera ontology-merging environment (McGuinness et al. 2000), an interactive merging tool based on the Ontolingua ontology editor (Farquhar et al. 1996), considers limited ontology structure in suggesting merging steps. However, the only relations that Chimaera currently considers is the subclass–superclass relation and slot attachment.

In our earlier work, we developed PROMPT—a tool for semi-automatic guided ontology merging (Noy and Musen 2000). PROMPT identifies candidates for merging as pairs of **matching terms**—terms from different source ontologies representing similar concepts. It determines not only syntactic but also semantic match based on (1) the content and structure of the source ontologies (e.g., names of classes and slots, subclasses, superclasses, domains and ranges of slot values) and (2) the user's actions (i.e., incorporating in its analysis the knowledge about similarities and differences that the user has already identified).

To summarize, those automatic approaches to semantic matching that do consider the structural relations among terms, base their analysis on studying only the terms that are *directly* related to one another. Both PROMPT and Chimaera consider subclasses and superclasses in question and slots directly attached to a class. PROMPT also considers classes that are referenced by the slots attached to the class in question.

Anchor-PROMPT, which we present here, complements these approaches by analyzing non-local context, by "looking further," and by providing additional suggestions for possible matching terms.

3 The Problem

To illustrate how Anchor-PROMPT works, we will consider two ontologies for representing clinical trials, their protocols, applications, and results. The first ontology, the Design-a-Trial (DaT) ontology (Modgil et al. 2000), underlies a knowledge-based system that helps doctors produce protocols for randomized controlled trials. The

second ontology, the randomized clinical-trial (RCT) ontology (Sim 1997), is used in creating electronic trial banks that store the results of clinical trials and allow researchers to find, appraise, and apply the results. Both ontologies represent clinical trials, but one of them, DaT, concentrates on defining a trial protocol itself, and the other, RCT, on representing the results of the trial. The two groups developed their respective ontologies completely independent from each other. Therefore there is no intensional correlation between them. As part of the work on representing clinical guidelines in our laboratory, we needed to merge the two ontologies.

We implemented Anchor-PROMPT based on the knowledge model defined by the Open Knowledge-Base Connectivity (OKBC) protocol (Chaudhri et al. 1998). An ontology in OKBC consists of classes organized in a hierarchy, instances of classes, and slots representing relations between classes and between instances of classes.

In Anchor-PROMPT, we represent classes, slots, and their relations in the ontologies as directed labeled graphs. Figure 2 shows a part of the graph representing the RCT ontology. Classes are **nodes** in the graph. Slots are **edges** in the graph. A slot S connects two classes, A and B, in the graph, if both of the following conditions are true:

(1) The slot S is attached to class A (either as template slot or as an own slot), and
(2) The class B is either a value of slot S for the class A, or B is the range of allowed values for slot S at class A.

For example, the edge representing the slot `latest-protocol` in the RCT ontology links the class `TRIAL` to the class `PROTOCOL` (Figure 2). The slot `latest-protocol` at class `TRIAL` can have as its values instances of the class `PROTOCOL`.

Two nodes connected by an edge in a graph are **adjacent**. There is a **path** between two nodes of a graph, A and B, if, starting at node A, it is possible to follow a sequence of adjacent edges to reach node B. The **length** of the path is the number of edges in the path.

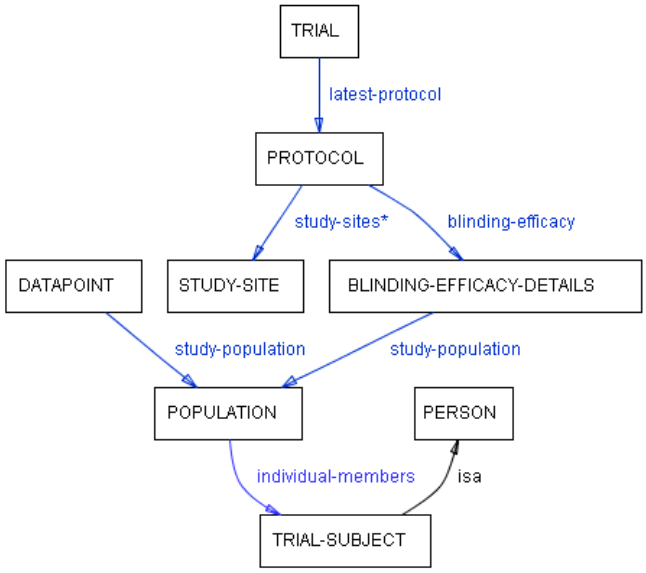


Figure 2. A graph representing a part of the RCT ontology.

The goal of the Anchor-PROMPT algorithm is to produce automatically a set of semantically related concepts from the source ontologies using a set of anchor matches identified earlier (manually or automatically) as its input.

4 The Anchor-PROMPT Algorithm

Anchor-PROMPT takes as input a set of **anchors**—pairs of related terms in the two ontologies. We can use any of the existing approaches to term matching to identify the anchors (Section 2). A user can identify the anchors manually. An automated system can identify them by comparing the names of the terms. For example, we can assume that if the source ontologies cover the same domain, the terms with the same names are likely to represent the same concepts. We can also use a combination of system-determined and user-defined anchors. We can use pairs of related terms that Anchor-PROMPT has identified in an earlier iteration after the user has validated them.

For the example in this section, we will consider the following two pairs of anchors for the two clinical-trial ontologies (the first class in the pair is in the RCT ontology; the second is in the DaT ontology[1]):

`TRIAL, Trial`
`PERSON, Person`

Using these two pairs as input, the algorithm must determine pairs of other related terms in the RCT and DaT ontologies. It generates a set of all the paths between `PERSON` and `TRIAL` in the RCT ontology and between `Person` and `Trial` in DaT ontology (Figure 3 shows some of these paths[2]). It considers only the paths that are shorter than a pre-defined parameter *length*. Now consider a pair of paths in this set that have the same length. For example:

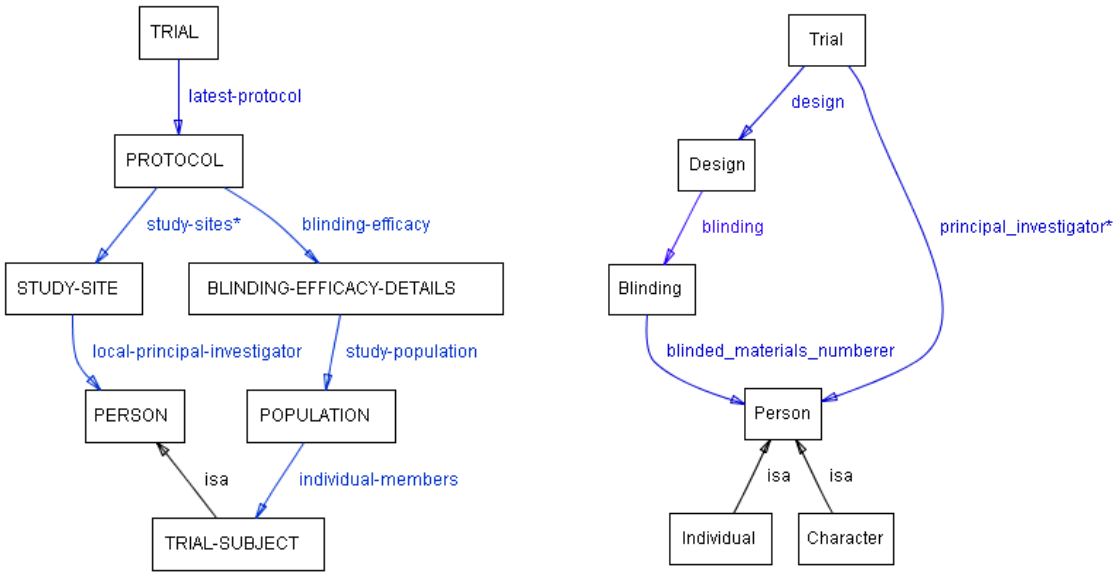
Path 1 (in the RCT ontology):
`TRIAL→PROTOCOL→STUDY-SITE→PERSON`
Path 2 (in the DaT ontology):
`Trial→Design→Blinding→Person`

As it traverses the two paths, Anchor-PROMPT increases the **similarity score**—a coefficient that indicates how closely two terms are related—for the pairs of terms in the same positions in the paths. For the two paths in our example, it will increase the similarity score for the following two pairs of terms:

`PROTOCOL, Design`
`STUDY-SITE, Blinding`

[1] The RCT ontology uses all `UPPER-CASE` letters for class names. The DaT ontology `Capitalizes` the class names. Therefore, it is easy to distinguish which class names come from which ontology, and we will sometimes omit the source information.

[2] We have changed the original RCT ontology slightly to simplify this example.



(a) (b)

Figure 3. (a) The paths between the classes `TRIAL` *and* `PERSON` *in the RCT ontology; (b) the paths between the classes* `Trial` *and* `Person` *in the DaT ontology*

Anchor-PROMPT repeats the process for each pair of paths of the same lengths that have one pair of anchors as their originating points (e.g., `TRIAL` and `Trial`) and another pair of anchors as terminating points (e.g., `PERSON` and `Person`). During this process it increases the similarity scores for the pairs of terms that it encounters. It aggregates the similarity score from all the traversals to generate the final similarity score. Consequently, the terms that often appear in the same positions on the paths going from one pair of anchors to another will get the highest score.

4.1 Equivalence groups

In traversing the graph representing the ontology and generating the paths between classes Anchor-PROMPT treats the subclass–superclass links differently from links representing other slots. Consider for example the path from `TRIAL` to `CROSSOVER` in Figure 4.

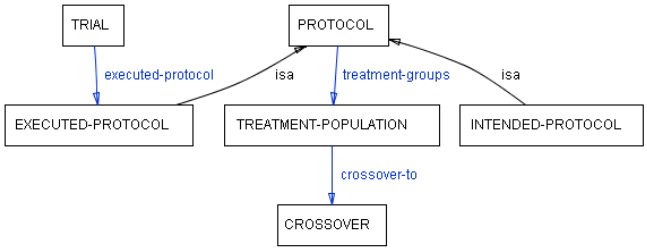


Figure 4. A path from `TRIAL` *to* `CROSSOVER`*: The classes* `EXECUTED-PROTOCOL` *and* `PROTOCOL` *form an equivalence group*

We could treat the `is-a` link in exactly the same way we treat other slots. However, this approach would disregard the distinct semantics associated with `is-a` links. Instead we can employ the difference in meaning between the `is-a` link and regular slots to improve the algorithm. An `is-a` link connects the terms that are already *similar* (e.g., `PROTOCOL` and `EXECUTED-PROTOCOL`); in fact one describes a *subset* of the other. Other slots link terms that are arbitrarily related to each other.

Anchor-PROMPT joins the terms linked by the subclass–superclass relation in **equivalence groups**. In the example in Figure 4, the classes `PROTOCOL` and `EXECUTED-PROTOCOL` constitute an equivalence group. Here is one of the paths from `TRIAL` to `CROSSOVER` in Figure 4 that goes through `EXECUTED-PROTOCOL`. We identify the equivalence group by brackets.

`TRIAL`→
`[EXECUTED-PROTOCOL, PROTOCOL]`→
`TREATMENT-POPULATION`→`CROSSOVER`.

Anchor-PROMPT treats an equivalence group as a single node in the path. The set of incoming edges for an equivalence-group node is the union of the sets of incoming edges for each of the group elements. Similarly, the set of outgoing edges for an equivalence-group node is the union of the sets of outgoing edges for each of its elements.

The `[EXECUTED-PROTOCOL, PROTOCOL]` equivalence group in Figure 4 has one incoming edge, `executed-protocol`, and one outgoing edge, `treatment-groups`.

4.2 Similarity score

We use the following process to compute the similarity score $S(C_1, C_2)$ between two terms C_1 and C_2 (where C_1 is a class from the source ontology O_1 and C_2 is a class from the source ontology O_2).

1. Generate a set of all paths of length less than a parameter L that connect input anchors in O_1 and O_2.
2. From the set of paths generated in step 1, generate a set of all possible pairs of paths of equal length such that one path in the pair comes from O_1 and the other path comes from O_2.
3. For each pair of paths in the set generated in step 2 and for each pair of nodes N_1 and N_2 located in the identical positions in the paths, increment the similarity score between each pair of classes C_1 and C_2 in N_1 and N_2 respectively be a constant X. (Recall that N_1 and N_2 can be either single classes or equivalence groups that include several classes).

Therefore the similarity score $S(C_1, C_2)$ is a **cumulative** score reflecting how often C_1 and C_2 appear in identical positions along the paths considering all the possible paths between anchors (of length less than L).

We change the constant by which we increment the similarity score when the matching nodes along the paths include not single classes but equivalence groups. Suppose we have the following two nodes at the same position on two paths between anchors: A_1 and $[B_2, C_2]$, a single class A_1 on one side, and an equivalence group with two classes B_2 and C_2 on the other side. Do we give the same score to both pairs of classes A_1, B_2 and A_1, C_2? Is this score the same as the one we would have given the pair A_1, B_2 had B_2 been the only class at the node? Do we give the pairs A_1, B_2 and A_1, C_2 any similarity score at all? We analyze the results for different values of these constants in Section 5.3.2.

4.3 Revisiting the example

We provided Anchor-PROMPT with the following set of three pairs of anchors from the RCT and DaT ontologies correspondingly:

```
TRIAL, Trial
PERSON, Person
CROSSOVER, Crossover
```

We allowed the paths of length less than or equal to 5 and limited the equivalence-group size to 2. Here are the output results in the order of the descending similarity score.

```
PROTOCOL, Design
TRIAL-SUBJECT, Person
INVESTIGATORS, Person
POPULATION, Action_Spec
PERSON, Character
TREATMENT-POPULATION, Crossover_arm
```

In fact, all but one of these results represents a pair of concepts that either are similar or one is a specialization (subclass) of the other. The only exception is the pair `POPULATION, Action_Spec`. Note that many of these pairings are specific to the domain of clinical trials (e.g., `PROTOCOL, Design` and `TRIAL-SUBJECT, Person`). The pair `PERSON, Character` indeed identifies the correct sense in which `Character` is used in the DaT ontology.

5 Evaluation

We perform a formative evaluation of Anchor-PROMPT by testing it on a pair of ontologies that were also developed independently by different groups of researchers. In our experiments, we varied the set of anchor pairs that was the input to the algorithm and various parameters, such as maximum path length, maximum size of equivalence groups, and constants in the similarity score computation. We then analyzed which fraction of the results produced by Anchor-PROMPT were indeed correct results and which parameter settings produced optimal performance.

5.1 Source ontologies

In order to evaluate Anchor-PROMPT formally, we chose a set of ontologies that was different from the two ontologies we used to develop and illustrate the algorithm. We imported two ontologies from the DAML ontology library (DAML 2001):

1. An ontology for describing individuals, computer-science academic departments, universities, and activities that occur at them developed at the University of Maryland (UMD), and
2. An ontology for describing employees in an academic institutions, publications, and relationships among research groups and projects developed at Carnegie Mellon University (CMU).[1]

These two ontologies constituted a good target for the merging experiment because on the one hand, they covered similar subject domains (research organizations and projects, publications, etc.) and on the other hand, their developers worked completely independent of each other and therefore there was no intensional correlation among terms in the ontologies.

5.2 Experiment setup

Input:

The UMD and the CMU ontologies;
Four anchor pairs.

Parameters:

1. A set of *anchor pairs* (we generated all possible sets of anchor pairs from the four input pairs)
2. The maximum number of elements allowed in an *equivalence group* (0, 1, or 2)
3. Similarity score for equivalence-group members along the path given that the score for single elements is 1 (1 and 3)
4. *Length* of path to consider (2, 3, or 4)

Output:

[1] Both ontologies consisted of several smaller ontologies which we merged into a single ontology for the experiment.

For each set of parameters, a set of related terms as determined by Anchor-PROMPT.

Process:

Run Anchor-PROMPT for all the possible combinations of parameters.

For each set of results, compute the *median* similarity score M and discard from the results set all pairs of terms with a similarity score less than M.

We then analyzed the results determining how many of the results were pairs of concepts that were either equivalent or were in a subclass–superclass relationship.

5.3 Evaluation results

5.3.1 Equivalence-group size

If the maximum equivalence-group size is 0 (do not consider subclass–superclass relationships at all) or 1 (allow equivalence groups of size 1), 87% of the experiments produce empty result sets. If the maximum equivalence-group size is 2, only 12% of the result sets are empty.

For the rest of the experiments we fix the maximum equivalence-group size at 2.

5.3.2 Similarity score for equivalence-group members

We have conducted two sets of experiments: in the first set all classes in the same position along the path got the same score N and in the second experiment classes that shared their position with other members of an equivalence group received only 1/3 of the score.[1]

Differentiating the score improved the correctness of results by 14%.

For the rest of the experiments we reduced the scores for members of equivalence groups.

5.3.3 Number of anchor pairs and maximum length of path

Table 1 presents results for various values for the two remaining parameters: the number of anchor pairs that were input to an experiment and the maximum allowed length of the path. For these experiments (as well as for a set of other experiments with different source ontologies), we received the best result precision (the highest ratio of correct results to all the returned results) with the maximum length of path equal to 2. When we limit the maximum path length to 3, we achieve the average precision of 61%. The precision goes up slightly (to 65%) with maximum path length of 4.

[1] In fact, varying the fraction of the score that we assigned to equivalence-group members has not changed the result: The results were identical for equivalence-group scores that were 1/3 or 1/2 of the score for single classes.

Max path length	Number of anchors	Result precision
4	4	67%
4	3	67%
4	2	61%
3	4	67%
3	3	61%
3	2	56%
2	4	100%
2	3	100%
2	2	100%

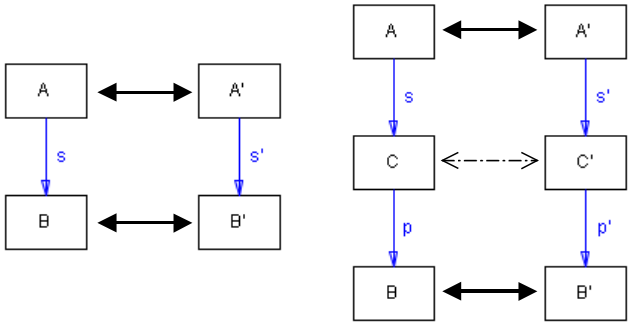
Table 1. Result precision with respect to maximum path length and the number of anchors.

6 Discussion

To understand the intuition behind Anchor-PROMPT, consider paths of length one (Figure 5a). Recall that the length of a path is the number of edges in the path. If class A is similar to class A' and class B is similar to class B', it is plausible to assume that the slots connecting them, s and s', are similar as well. In Figure 5b, we introduce an additional class, C and C' correspondingly, on the path. We get the paths of length 2). We continue the analogy by assuming that there is an increased likelihood that C and C' are similar. In addition, slots s and s' and p and p' are similar (Anchor-PROMPT does not currently record the similarity among slots).

The algorithm is based on the assumption that developers link the terms in the ontology in a similar manner even if they do not call the terms with the same names. Therefore, very long paths are unlikely to produce accurate results. As the path that we traverse becomes longer, it becomes less likely that they represent the same series of terms and relations.

Very short paths, however, consistently produced extremely small (but also extremely precise results sets). For the maximum path length of 2, Anchor-PROMPT produced result sets that contained only one pair of terms (with a similarity score above the median for that set) but this pair was always a correct one.



(a) (b)

Figure 5. The simple case: the paths of length 1 and 2.

Setting maximum path length to 0 will produce the results that are equivalent to Chimaera's results. Limiting the path length by 1 will produce the results that are equivalent to PROMPT's results (Section 2).

6.1 Reducing the effect of negative results

The similarity score between concepts is a *cumulative* similarity score: Anchor-PROMPT combines the score along all the paths. As a result, we reduce the effect of false matches: Two unrelated terms could certainly appear in identical positions in one pair of paths (and usually do). However, the same two unrelated terms are less likely to appear in identical positions on a different pair of paths.

To remove these incidental matches, we determine the median similarity score in each experiment and discard the pairs of terms with a similarity score less than the median. Therefore, Anchor-PROMPT will discard most of the incidental pairs of terms because they would have appeared only once in the identical positions and would have a low similarity score.

6.2 Performing ontology mapping

Throughout our discussion we have referred to the process of ontology merging, the process in which we start with two source ontologies and generate a new ontology that includes and reconciles all the information from the two source ontologies.

However, the approach that we have presented can be used directly for creating a mapping between terms in ontologies, as well as in matching database schemas. The result of the Anchor-PROMPT algorithm is a set of pairs of similar terms ranked by how close to each other the terms are. This result can be used either to trigger merging of the closely related terms or to establish a mapping between the terms.

6.3 Limitations

Anchor-PROMPT produced highly promising results with two sets of ontologies that were developed entirely independently from each other.

Our approach does not work equally well for all ontologies however. The approach does not work well when the source ontologies are constructed differently. For example, we used Anchor-PROMPT to find related terms in two ontologies of problem-solving methods: (1) the ontology for the unified problem-solving method (UPML) development language (Fensel et al. 1999) and (2) the ontology for the method-description language (MDL) (Gennari et al. 1998). Both ontologies describe reusable problem-solving methods, however, their designers used different approaches to knowledge modeling. The UPML ontology has a large number of classes with slots attached to and referring to classes at many different levels of hierarchy. The MDL ontology has a lot fewer classes with the hierarchy which is only two levels deep. If we think of the ontologies in terms of a graph, many of the nodes from the UPML ontology were "collapsed" in a single node in the MDL ontology. As a result, no two pairs of anchors had paths with the same length between them and the output of Anchor-PROMPT was empty.

In general, Anchor-PROMPT does not work well when one of the source ontologies is a deep one with many inter-linked classes and the other ontology is a shallow one where the hierarchy has only a few levels and most of the slots are associated with the concepts at the top of the hierarchy, and very few notions are reified. If this is the case, the results produced by the algorithm are no different from the results produced by the approaches that consider only very local context.

7 Conclusions

The Anchor-PROMPT algorithm that we have presented uses relations among the terms in an ontology and a set of anchors—pairs of similar terms—to determine which other terms in the ontology are similar.

We conducted experiments using unrelated source ontologies developed by different research groups. We have achieved the results that could not have been achieved using just the terms names (e.g., determine that `TRIAL-SUBJECT` and `Person` are very similar terms in the ontology of trial protocols).

Our experiments show that we can achieve result precision between 61% and 100% depending on the size of the initial anchor set and the maximum length of the path that we traverse.

The algorithm relies on *limited* input from the user. The user does not need to analyze the structure of the ontology deeply, just to determine some pairs of terms that "look similar".

Based on our results, we believe that Anchor-PROMPT can significantly improve the sets of suggestions that other tools identify by producing sets of semantically similar terms using a small set of previously determined similar terms.

Acknowledgments

We have implemented Anchor-PROMPT as a plugin to the Protégé-2000 ontology-editing and knowledge-acquisition tool developed at Stanford Medical Informatics (http://protege.stanford.edu). We generated all the graphs in this paper automatically using OntoViz, a Protégé-2000 plugin, developed by Michael Sintek. Samson Tu and John Nguyen helped us to understand the clinical-trial ontologies. Whatever clarity and readability this paper has, it owes it all to the detailed and thoughtful comments of Monica Crubézy and Ray Fergerson. We are very grateful to anonymous reviewers for suggestions on improving the paper. This work was supported in part by a grant from Spawar and by a grant from FastTrack Systems, Inc

References

Brickley, D. and Guha, R.V. (1999). Resource Description Framework (RDF) Schema Specification. Proposed

Recommendation, World Wide Web Consortium**:** http://www.w3.org/TR/PR-rdf-schema.

Castano, S. and De Antonellis, V. (1999). A Schema Analysis amd Reconciliation Tool Environment. In: *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'99)*, IEEE.

Chapulsky, H., Hovy, E. and Russ, T. (1997). Progress on an Automatic Ontology Alignment Methodology.

Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P. (1998). OKBC: A programmatic foundation for knowledge base interoperability. In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, Wisconsin, AAAI Press/The MIT Press.

DAML (2001). DAML ontology library. http://www.daml.org/ontologies/

Farquhar, A., Fikes, R. and Rice, J. (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction. In: *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Fensel, D., Benjamins, V.R., Motta, E. and Wielinga, R. (1999). UPML: A Framework for knowledge system reuse. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden.

Gennari, J.H., Grosso, W. and Musen, M.A. (1998). A method-description language: An initial ontology with examples. In: *Proceedings of the Eleventh Banff Knowledge Acquisition for Knowledge-Bases Systems Workshop*, Banff, Canada.

Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**: 199-220.

Humphreys, B.L. and Lindberg, D.A.B. (1993). The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association* **81**(2): 170.

McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S. (2000). An Environment for Merging and Testing Large Ontologies. *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*. A. G. Cohn, F. Giunchiglia and B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers.

Milo, T. and Zohar, S. (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. In: *Proceedings of the 24th International Conference on Very Large Data Bases*, New York City, Morgan Kaufmann.

Modgil, S., Hammond, P., Wyatt, J. and Potts, H. (2000). The Design-A-Trial Project: Developing A Knowledge-Based Tool for Authoring Clinical Trial Protocols. In: *Proceedings of the First European Workshop on Computer-based Support for Clinical Guidelines and Protocols (EWGLP 2000)*, Leipzig, Germany, IOS Press, Amsterdam.

Musen, M.A. (1992). Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* **25**: 435-467.

Netscape (1999). DMOZ Open Directory. http://www.dmoz.org/

Noy, N.F. and Musen, M.A. (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX.

Price, C. and Spackman, K. (2000). SNOMED clinical terms. *BJHC&IM-British Journal of Healthcare Computing & Information Management* **17**(3): 27-31.

Sim, I. (1997). Trial Banks: An Informatics Foundation for Evidence-Based Medicine. PhD Dissertation, Stanford University**:** SMI-97-0701/STAN-CS-TR-97-1599.

Wiederhold, G. and Jannink, J. (1999). Composing Diverse Ontologies. In: *Proceedings of the IFIP Working Group on Database, 8th Working Conference on Database Semantics (DS-8)*, Rotorua, New Zealand.

Ontology Integration: How to perform the Process*

Helena Sofia Pinto and **João P. Martins**
Grupo de Inteligência Artificial
Departamento de Eng. Informática
Instituto Superior Técnico
Universidade Técnica de Lisboa
Av. Rovisco Pais
1049-001 Lisboa, Portugal

Abstract

Although ontology reuse is an important research issue only one of its subprocesses (merge) is fairly well understood. The time has come to change the current state of affairs with the other reuse subprocess: integration. In this paper we characterize the ontology integration process, we identify the activities that should be performed in this process and describe a methodology to perform the ontology integration process.

1 Introduction and motivation

Ontologies aim at capturing static domain knowledge in a generic way and provide a commonly agreed upon understanding of that domain, which may be reused and shared across applications and groups [Chandrasekaran *et al.*, 1999]. Therefore, one can define an ontology as a shared specification of a conceptualization. Ontology reuse is now one of the important research issues in the ontology field. There are two different reuse processes [Pinto *et al.*, 1999]: (1) *merge* and (2) *integration*. Merge is the process of building an ontology in one subject reusing two or more different ontologies on that subject [Pinto *et al.*, 1999]. In a merge process the source ontologies are unified into a single one, so it usually is difficult to identify regions in the resulting ontology that were taken from the merged ontologies and that were left more or less unchanged.[1] It should be stressed that in a merge process the source ontologies are truly different ontologies and not simple revisions, improvements or variations of the same ontology. Integration is the process of building an ontology in one subject reusing one or more ontologies in different subjects[2] [Pinto *et al.*, 1999]. In an integration process source ontologies are aggregated, combined, assembled together, to form the resulting ontology, possibly after reused ontologies have suffered some changes, such as, extension, specialization or adaptation. In an integration process one can identify in the resulting ontology regions that were taken from the integrated ontologies. Knowledge in those regions was left more or less unchanged.

A lot of research work has been conducted under the merge area. There is a clear definition of the merge process [Sowa, 2000], operations to perform merge have been proposed [Noy and Musen, 1999; Wiederhold, 1994], a methodology is available [Gangemi *et al.*, 1998] and several ontologies have been built by merging several ontologies into a single one that unifies all of the reused ontologies [Swartout *et al.*, 1997; Gangemi *et al.*, 1998]. The first tools to help in the merge process are now available [Noy and Musen, 2000; McGuiness *et al.*, 2000].

In the integration area a similar effort is now beginning. The most representative ontology building methodologies [Uschold and King, 1995; Gruninger, 1996; Fernández *et al.*, 1999] recognize integration as part of the ontology development process, but none really addresses integration. Integration is only recognized as a difficult problem to be solved. They don't even agree on what integration is: for some it is an activity, for others it is a step. We have been involved in two integration experiences where publicly available ontologies were reused: we built the Reference ontology [Arpirez-Vega *et al.*, 2000; Pinto and Martins, 2000; Pinto, 1999a; Arpirez-Vega *et al.*, 1998] and we were involved in building some of the subontologies needed to build an Environmental Pollutants ontology (EPO) [Pinto and Martins, 2000; Pinto, 1999a; Amaya, 1998; Gómez-Pérez and Rojas-Amaya, 1999].

We have found that integration is far more complex than previously hinted. It is a process of its own [Pinto, 1999a; Pinto and Martins, 2000]. In this article we characterize integration, we identify the activities that should be performed in this process and we characterize those activities. We describe the methodology that we developed to perform the activities that form this process.

2 Terminology and assumptions

Ontology building is a process that follows an evolving prototyping life cycle. The usually accepted *stages* through which

*This work was partially supported by JNICT grant No. PRAXIS XXI/BD/11202/97 (Sub-Programa Ciência e Tecnologia do Segundo Quadro Comunitário de Apoio).

[1] In some cases, knowledge from the merged ontologies is homogenized and altered through the influence of one source ontology on another (is spite of the fact that the source ontologies do influence the knowledge represented in the resulting ontology). In other cases, the knowledge from one particular source ontology is scattered and mingled with the knowledge that comes from the other sources.

[2] The subjects of the different ontologies may be related.

an ontology is built are:[3] specification, conceptualization, formalization, implementation, and maintenance. At each stage there are *activities* to be performed. Besides the activities of *specification*, in which one identifies the purpose (why is the ontology being built?) and scope (what are its intended uses and end-users?) of the ontology, *conceptualization*, in which one describes, at a conceptual level, the ontology that should be built so that it meets the specification found in the previous step, *formalization*, in which one transforms the conceptual description into a formal model, *implementation* in which one implements the formalized ontology in a formal knowledge representation language, and *maintenance*, in which one updates and corrects the implemented ontology; that should be performed at each homonymous stage, there are other activities, such as, *knowledge acquisition*, in which one acquires knowledge about the domain either by using elicitation techniques on domain experts or by referring to relevant bibliography, *documentation*, in which one reports in a document and along the implementation, what was done, how it was done and why it was done, *integration*, in which one reuses other ontologies as much as possible, and *evaluation*, in which one technically judges the ontology.

For us, an ontology consists of: classes, instances, relations, functions and axioms. Generically, we refer the union of classes and instances as *concepts*. Each one of the constituents of an ontology is generically referred to as a *knowledge piece*. Each knowledge piece is associated with a name, a documentation and a definition.

The aim of the conceptualization phase is to describe in a conceptual model the ontology that should be built. We assume that, in this phase of *any* ontology building process questions like,

- what should be represented in the ontology?
- how should it be represented (as a class, relation, etc.)?
- which relation should be used to structure knowledge in the ontology?
- which structure is the ontology going to have (graph, tree, etc.)?
- which ontological commitments and assumptions should the ontology comply to?
- which knowledge representation ontology should be used?
- should the ontology be divided in modules?
- in which modules should the ontology be divided?

are answered.

3 The Process

In this section we present the most important conclusions about integration and its characterization.

[3]We use the terminology proposed in [Fernández *et al.*, 1999] since it is the most consensual in the field.

3.1 Main findings

The main conclusion is that integration is a process that takes place along the entire ontology building life cycle, rather than a step or an activity, as previous ontology building methodologies proposed [Pinto, 1999a; Pinto and Martins, 2000].

As any process, integration is composed of several activities. We have identified the activities that should take place along the ontology building life cycle to perform integration. Since the development of an ontology follows an evolving prototyping life cycle, integration activities can take place for one ontology in any stage of the ontology building process.

Another important conclusion is that integration should begin as early as possible in the ontology building life cycle so that the overall ontology building process is simplified [Pinto, 1999a; Pinto and Martins, 2000]. In both our cases, integration began as early as the conceptualization phase. Since in conceptualization much of the design of the ontology is specified, it is considerably more difficult to try to integrate an ontology at the implementation phase because, unless one has prior knowledge of the ontologies available for reuse, available ontologies will rarely match the needs and the conceptual model found for the resulting ontology. One of the consequences of this conclusion is that more integration effort should be made at the earliest stages, specially in conceptualization and formalization, than at final ones, implementation or maintenance [Pinto and Martins, 2000].

At the conceptualization phase, one uses knowledge level [Newell, 1982] representations of ontologies. Usually, the knowledge level representation of an ontology is not publicly available (only implemented ontologies are available at ontology libraries). If the knowledge level representation of an ontology is not available, then an ontological reengineering process [Blázquez *et al.*, 1998] can be applied to get the conceptual model of an implemented ontology. This process returns one possible[4] conceptual model of an implemented ontology. When one begins integration as early as conceptualization, one needs the ontologies that are going to be considered for integration represented in an adequate form. Any conceptual model representation is adequate. An important point to be stressed out from all of our experiences is the fact that we had access to knowledge level representations of most reused ontologies as proposed by METHONTOLOGY [Fernández *et al.*, 1997]. In the case of $(KA)^2$ [Benjamins and Fensel, 1998; Benjamins *et al.*, 1999] (to build the Reference ontology) and Chemicals [Gómez-Pérez *et al.*, 1996; Fernández *et al.*, 1999] (to build the Monoatomic Ions subontology of EPO) we had access to the actual conceptual models that produced their Ontolingua versions, but, in the case of EPO a reengineering process was applied [Gómez-Pérez and Rojas-Amaya, 1999] to produce one conceptual model of Standard Units [Gruber and Olsen, 1994]. However, any knowledge level representation would be appropriate. Moreover, due to the particular framework that was used, ODE [Fernández *et al.*, 1999], all

[4]It should be stressed that this process may not produce the actual conceptual model that originated the final ontology. Moreover, if the conceptual model found for the ontology after the reverse engineering step shows some deficiencies, it may be improved through a restructuring step.

of our work was done at the knowledge level. This simplified the overall process of integration a lot.

We would also like to point out that in both cases there was no need to translate ontologies between different knowledge representation languages. Translation of ontologies is in itself a very important and difficult problem to be solved in order to allow more generalized reuse of ontologies. As discussed in [Uschold *et al.*, 1998; Russ *et al.*, 1999], translation is far from being a fully automatic process in the near future.

3.2 Integration activities

We are going to describe the most important activities that compose the ontology integration process. All integration activities assume that the ontology building activities are also performed, that is, the integration process does not substitute the ontology building process, it rather is a part of it.

Identify the possibility of integration The framework being used to build the ontology should allow some kind of knowledge reuse. For instance, the Ontolingua Server [Farquhar *et al.*, 1996] maintains an ontology library and allows integration operations, such as inclusion or restriction. More general systems, such as KACTUS do not allow such kind of operations, but allow pre-existent ontologies to be imported and edited. In other cases, integration (or any kind of reuse) may involve rebuilding an ontology in a framework different from the one where the ontology is available. In some cases, this may be cost-effective, in others it may be more cost-effective to build a new ontology from scratch that perfectly meets present needs and purposes than to try to rebuild and adapt a pre-existent ontology.

Identify the modules in which the ontology can be divided into The modules (building blocks) needed to build the future ontology are identified, that is, in which subontologies should the future ontology be divided (in integration, the modules are obviously related to ontologies). Upper-level modules and domain modules have to be identified. [5]

Identify the assumptions and ontological commitments that each module should comply to The assumptions and ontological commitments [Gruber, 1995] are described in the conceptual model and in the specification requirements document of the future ontology. This is one of the activities where documentation of an ontology can be crucial to allow better, faster and easier reuse. The assumptions and ontological commitments of the building blocks should be compatible among themselves and should be compatible with the assumptions and ontological commitments found for the resulting ontology.

Identify what knowledge should be represented in each module At this stage, one is only trying to have an idea of what the modules that are going to compose the future ontology should "look like" in order to recognize whether available ontologies are adequate to be reused. At this stage one only identifies a list of essential concepts. The conceptual model of the ontology and abstraction capabilities are used to produce such list. [6]

Identify candidate ontologies that could be used as modules This is subdivided into: (1) *finding available ontologies*, and (2) *choosing from the available ontologies which ones are possible candidates to be integrated*. To find possible ontologies one uses ontology sources. Since available ontologies are mainly implemented ones one should look for them in ontology libraries, as for instance, in the Ontolingua Server [7] for ontologies written in Ontolingua, in Ontosaurus [8] [Swartout *et al.*, 1997] for ontologies implemented in Loom [MacGregor, 1990a], or in the Cyc Server [9] for Cyc's upper-level ontology. Conceptualized or formalized ontologies are more difficult to find. Sometimes they are available in the literature or can be obtained by contacting ontology builders. However, not every ontology in a given subject will be appropriate to be reused. Some may lack some important concepts, etc. Therefore, from the available ontologies, one must chose those that satisfy a series of requirements. In the next section we discuss in detail how this choice is performed.

Get candidate ontologies in an adequate form This includes, not only, its knowledge level or implementation level *representations*, but also, all available *documentation*. As already discussed, one should prefer to work with the knowledge level representation of an ontology, if available. In some cases, this representation can be found in the literature (technical reports, books, thesis, etc.), or at least parts of it. Another possibility is contact ontology developers. However, in most cases, only the implementation level representation of an ontology is available, or is more easily available. Therefore, the *reengineering process* may be applied using the particular framework that was adopted to design the resulting ontology. If the ontology is not available (either at the implementation or knowledge level), one can still try to reconstruct it, or, at least, parts of it, using available documentation. While getting the implementation level representation of an ontology, if the ontology is not written in the adequate language (the language that was chosen to represent the resulting ontology) a knowledge *translation process* must take place. There are only a few translation attempts. In general, there are not many translators available, their technology is still immature and improving existing translators is a rather difficult task. In [Uschold *et al.*, 1998] the translation was done by hand and the conclusion was that this process is far from being a fully automatic process in the near future. Automatic translators are still at draft level [Russ *et al.*, 1999], therefore a lot of human intervention is needed to improve ontology translated versions. If translators are available they should be used to produce initial versions. Then, these initial versions should be improved by hand. Translators between different knowledge level representation languages are currently not available. The translation process is, in general, complex. It is important that, if the ontology includes other ontologies, one should also get the included ontolo-

[5] Representation ontologies are chosen in any ontology building process. Therefore, they are not specifically addressed here.

[6] At later stages one will need to know to what level of detail should that knowledge be represented, which relations should organize (structure) the ontology, and it would be helpful to know how it should be represented (concept, relation, etc.).

[7] `http://WWW-KSL-SVC.stanford.edu:5915`

[8] `http://www.isi.edu/isd/ontosaurus.html`

[9] `http://www.cyc.com`

gies. When reusing/using one ontology one must understand it fully, which includes every definition of every knowledge piece represented in the ontology (directly or indirectly). Included ontologies are a hidden part of the ontology. Knowledge pieces from the included ontologies can be used in the definitions of the ontology, therefore, in order to understand the ontology and know what is meant by one knowledge piece that comes from an included ontology one must have access to it and its definition or its technical documentation.

Study and analysis of candidate ontologies This includes two important activities: (1) *technical evaluation* of the candidate ontologies by *domain experts* through specialized criteria *oriented to integration* and (2) *user assessment* of the candidate ontologies by *ontologists* through specialized criteria *oriented to integration*. The specialized criteria used in integration oriented evaluation and assessment enhance the possible problems that a particular ontology may have in a particular integration process. They allow ontologists and domain experts to identify and be aware of those problems. In the next section we discuss the criteria to be used.

Choosing the most adequate source ontologies to be reused At this stage, and given the study and analysis of candidate ontologies performed by domain experts and ontologists, the final choices must be made. Among the chosen candidate ontologies that were technically evaluated and user assessed for integration one has to choose the ontology (or set of ontologies) that best suit our needs and purpose, or that can more easily or better be adapted to them. The ontology(ies) chosen to be reused may lack knowledge, may require that some knowledge is removed, etc., that is, it(they) may not exactly be what is needed. The best candidate ontology is the one that can best (more closely) or more easily (using less operations) be adapted to become the needed ontology. This choice also depends to some extent on the other ontologies that are going to be reused since in an integration process one can reuse more than one ontology. It is important that reused ontologies are compatible among themselves, namely in what concerns the overall coherence. Sometimes, one can choose more than one ontology in a given subject if each one focuses different points of view of that subject. In the next section we go into the details of this choice.

Integrate knowledge All these activities precede integration of knowledge from the integrated ontology into the resulting ontology. They help the ontologist to analyze, compare, and choose the ontologies that are going to be reused. When this part of the process ends, that is the appropriate ontologies to be reused in one particular integration process are found, we must integrate the knowledge of those ontologies. For that, one needs *integration operations* and *integration oriented design criteria*. Integration operations specify how knowledge from an integrated ontology is going to be included and combined with knowledge in the resulting ontology, or modified before its inclusion. These can be viewed as composing, combining, modifying or assembling operations. Knowledge from integrated ontologies can be, among other things, (1) used as it is, (2) adapted (or modified), (3) specialized (leading to a more specific ontology on the same domain) or (4) augmented (either by more general knowledge or by knowledge at the same level). Design criteria guide the

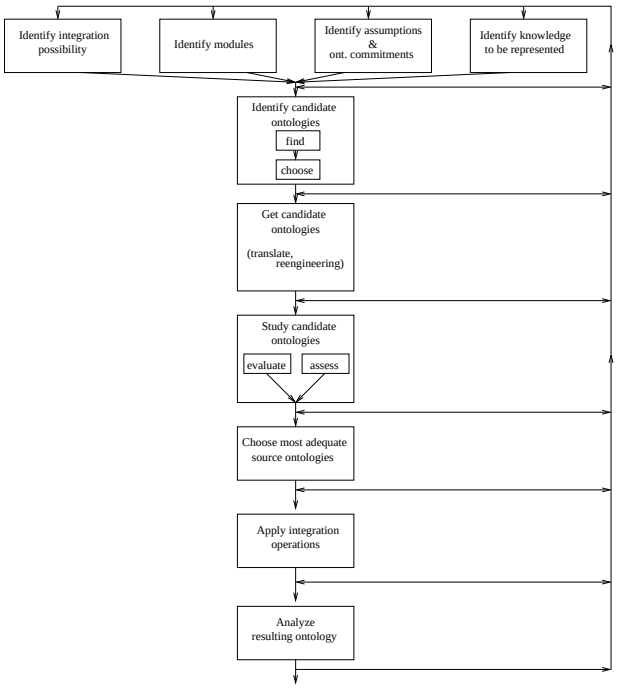


Figure 1: The integration process

application of integration operations so that the resulting ontology has an adequate design and is of quality. In the next section we discuss the integration operations that were found useful in our integration experiences and the design criteria that guided their application.

Analyze resulting ontology After integration of knowledge one should evaluate and analyze the resulting ontology. Besides the usual criteria involved in evaluation of any ontology [Gómez-Pérez *et al.*, 1995] and the features that any ontology with an adequate design should comply to [Gruber, 1995] one should pay attention to specialized criteria that specifically analyzes whether the resulting ontology has enough quality. They are discussed in the next section.

3.3 Discussion

In Figure 1 we present the activities that compose the ontology integration process. Although ontology building and consequently ontology integration follows an evolving prototyping life cycle, some order must be followed. In general, the activities that compose the integration process tend to be performed following the order by which they were presented. However, some of the activities (and subactivities) to be performed before applying integration operations are interchangeable and some may be even performed in parallel. For instance, integration-oriented technical evaluation and user assessment of candidate ontologies. Moreover, the auxiliary subprocesses, reengineering and translation, may not occur in a particular integration process. If we find an ontology that matches the whole ontology that one needs to build, then one does not need to apply integration operations or analyze the resulting ontology. However, finding candidate ontologies, their evaluation and assessment for integration purposes, and

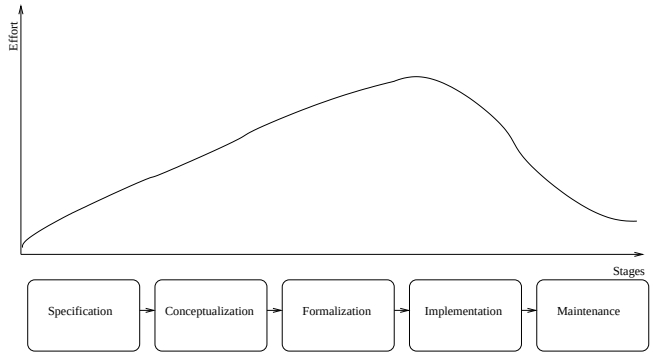


Figure 2: Integration effort along the ontology building process

the choice of the most adequate one remain essential activities to be performed. Finally, one can go back from any stage in the process to any other stage as entailed by the kind of life cycle. The important issue is that these activities are present in any integration process, although sometimes not explicitly or with different levels of importance and effort.

All activities, in particular those that precede application of integration operations, should be performed preferably in conceptualization or in formalization stages, that is, before implementation (some methodologies jump directly from conceptualization to implementation). However, if integration begins later in the ontology development life cycle, they still have to be performed. In both our integration experiences the framework that we used, ODE, automatically generated the implemented versions of the resulting ontologies. Therefore, we performed all integration activities during conceptualization and formalization stages. Using other frameworks may extend the process a bit. If the framework being used does not generate the implementation of the resulting ontology from the conceptual representations, after performing all activities at the knowledge level, the implemented versions of the chosen ontologies must be obtained and then one must apply the already determined sequence of integration operations in order to build the implemented version of the resulting ontology. In this case, only two activities (get ontologies and apply integration operations) had to be performed at the implementation level. This particular process falls into a typical evolving prototyping life cycle.

One important aspect of integration is the fact that this process is included in the overall ontology building process. The relation between the integration process and the overall ontology building process is shown in Figure 2. In the case that an ontology adequate to be reused is not found one must build it from scratch using one of the available ontology building methodologies.

The integration effort grows from specification and conceptualization to formalization where it reaches its maximum. It begins to decrease during implementation. It should be noted that in our particular case, due to the particular framework that was used the integration effort during implementation was null. The integration effort is not null during maintenance since integrated ontologies may themselves change due to maintenance activities making it necessary (or desirable) to reapply the integration process.

4 A Methodology

In this section we present the methods, procedures and guidelines that we developed to perform the activities that form this process. They form a methodology to perform integration.

4.1 Choosing candidate ontologies

To *choose candidate ontologies* one analyzes a series of features.[10] At this stage of the ontology integration process one is not going to be very particular, fussy, about the ontology, since one does not want to leave out any possible candidate. Therefore, only a very general analysis is made. Some of those features are *strict requirements*:

1. domain
2. is the ontology available?
3. formalism paradigms in which the ontology is available
4. main assumptions and ontological commitments
5. main concepts represented

If the ontology does not have adequate values for these properties they cannot be considered for integration. Therefore, these properties are used to eliminate ontologies. Other features are *desirable requirements* or desirable information:

1. where is the ontology available?
2. at what level is the ontology available?
3. what kind of documentation is available (technical reports, articles, etc.)?
4. where is that documentation available?

If some of the properties have certain values, the ontology is a better candidate: if the knowledge level representation of an ontology is available, then this ontology is a better candidate since the reengineering process would not have to be performed, if the internal and external documentation is available, then the most relevant information about the construction and choices made during the construction of the ontology is available, but if only articles are available about the ontology, then it is likely that some of the choices are not explained. If all of the values of these properties are unknown, then the ontology will not be a candidate, that is, if one cannot find where the ontology and the documentation is available, one cannot reuse it, therefore, the ontology is not a candidate. However, if there is enough documentation available, then it may be possible to reconstruct the ontology, and if the ontology is available, then it may be possible to understand it, provided that the domain is common enough and the ontology is simple and not very large (and possibly after some knowledge acquisition).

One can use a very simple metric to combine these different features. If strict requirements do not have adequate values, the ontology is eliminated. If desirable requirements

[10] Here we only describe the most important features involved in this choice. They are all organized into a taxonomy.

have appropriate values, then the ontology is a better candidate. If not, they are a worse candidate. If none of the desirable requirements have appropriate values, then the ontology is not a candidate. One does not want to eliminate any possible candidate at this stage of the integration process, only those that are of no use at all.

If, in a particular integration process, other features should be taken into consideration while choosing candidate ontologies, the metrics can be easily updated to take into account those new features. One only has to decide whether they are strict or desirable requirements The advantage of the flexibility of this metric is the fact that it can be better adapted to integration processes that should take into account particular features during the choice of one ontology. In particular, this kind of changes can narrow down the possible ontologies to choose from, if one introduces more strict requirements. For instance, one can impose the condition that only already evaluated ontologies should be considered as candidates. In that case, one should add this feature as a strict requirement. If one only wishes to prefer already evaluated ontologies, then this feature should be added as a desirable requirement.

4.2 Study and analysis of candidate ontologies

To *technically evaluate candidate ontologies* the domain experts should analyze the ontology paying special attention to [Pinto, 1999a; Pinto and Martins, 2000]:

- what knowledge is missing (concepts, classification criteria, relations, etc),
- what knowledge should be removed,
- which knowledge should be relocated,
- which knowledge sources changes should be performed,
- which documentation changes should be performed,
- which terminology changes should be performed,
- which definition changes should be made,
- which practices changes should be made.

Since domain experts usually find the languages used to implement ontologies difficult to understand [Fernández *et al.*, 1999], they should preferably be given a knowledge level representation of the ontology.

To *user assess candidate ontologies* the ontologists should analyze the ontology paying special attention to [Pinto, 1999a; Pinto and Martins, 2000]:

- the overall structure of the ontology (one hierarchy, several hierarchies, a graph, etc.) to assess whether the ontology has an adequate (and preferably well-balanced) structure, adequate and enough modules, adequate and enough specialization of concepts, adequate and enough diversity, similar concepts are represented closer whereas less similar concepts are represented further apart, knowledge is correctly "placed" in the structure so that inheritance mechanisms can infer appropriate knowledge from the ontology, etc;
- the distinctions (classification criteria made of the concepts described in the ontology) upon which the ontology is built to assess whether they are relevant and exactly the ones (quantity and quality) required;
- the relation used to structure knowledge[11] in the ontology to assess whether it is the required one;
- the naming convention rules used to assess whether they ease and promote reuse;
- the quality of the definitions (do they follow unified patterns, are simple, clear, concise, consistent, complete, correct —lexically and syntactically—, precise and accurate);
- the quality of the documentation of the ontology,
- the knowledge pieces represented (or included) are the ones that should be represented and all appropriate knowledge pieces are represented, etc.

Both domain experts and ontologists should evaluate and assess all and the whole of possible candidate ontologies. In [Pinto and Martins, 2000] a detailed discussion about the sets of integration oriented evaluation and assessment criteria can be found.

4.3 Choosing source ontologies

Choosing source ontologies is a rather complex multi-criteria choice where a lot of different aspects are involved. It is a much more complex choice than choosing candidate ontologies. For this reason, we propose that the task of choosing source ontologies should be divided into *two stages*.

First stage

In the *first stage* one tries to find which candidate ontologies are best suited to be integrated. Domain expert and ontologist analyses are crucial in this process. We propose that candidate ontologies should be analyzed according to a taxonomy of features, Figure 3.

General features give general information about the ontology. It is important that the ontology is of an adequate type, (*general or domain*). Depending on the *formality* [Uschold and Gruninger, 1996] of the resulting ontology one may integrate different kinds of ontologies. *Development status* gives information about the degree of readiness of an ontology to be reused (intended, on-going, toy example, implemented, mature). A toy example will only have representative knowledge pieces represented. An implemented ontology can be a good candidate provided that it has been carefully built or it has been evaluated. A mature ontology used in applications is a good candidate. This ontology should be a more or less stable ontology (provided that the domain does not evolve very rapidly).

Development features are related to how the ontology was built. The *quality of knowledge sources* and *adequacy of knowledge acquisition practices* are analyzed during the domain expert integration-driven technical evaluation. It is important that the ontology is *maintained*. One interesting finding about ontologies is the fact that they evolve, are "living", since their domains also evolve. Therefore, if they are maintained, it is most likely that they are updated. If they are maintained, it is important to know how maintenance is performed. Maintenance policies differ in *who* changes the

[11] An ontology can be thought of as structured or organized according to one privileged relation, for example, ISA, part-of, etc.

- general
  - generality
  - formality
  - development status
- development
  - knowledge acquisition
    - quality of knowledge sources
    - adequacy of knowledge acquisition practices
  - maintenance
    - is it maintained?
    - who does maintenance?
    - how is maintenance done?
  - documentation
    - quality of the documentation available
    - is the available documentation complete?
  - implementation
    - language issues
      - language(s) in which it is available
      - translators: are there translators? for which languages? quality of those translators
      - properties needed of the KR system in which it is built
- content
  - level of detail
  - modularity
  - adequacy from the domain expert point of view
  - adequacy from the ontologist point of view

Figure 3: Features for choosing source ontologies, first stage

ontology (can anybody change the ontology, or only authorized personnel?) and *how* those changes are performed (is the ontology changed regardless of people that built it, use it or reuse it? are the suggestions of change previously discussed among those groups? is there any attempt to reach a consensus between those groups? is there a special board that decides upon suggestions for changes?). It is important that the *documentation* has enough *quality* (it is clear, it adequately describes the domain, the ontology, the alternative representations of that ontology and which alternatives were preferred) and is *complete* (the ontology is completely described).

The language in which the ontology is represented is a rather important issue. If the ontology is available in the required language the task is greatly simplified. Although translation of ontologies is an important activity in integration, the overall effort of building the ontology can be considerably lessened if we avoid it. Therefore, it is important to know in which *languages* the ontology is available, whether *translators* from those languages are available, *for which languages?* those translators are available and their *quality*. It is also important to know which *reasoning capabilities* are needed by the ontology from the knowledge representation system where it is implemented, in order to know whether the ontology can be represented under a different knowledge representation system. Even if translators are available, one may not be sure of the possibility of full translation between different knowledge representation systems. For instance, while translating an ontology represented in first order logic into a pure frame system, if axioms are represented, they are lost. Therefore, one needs to know, among other issues:

- *formalism paradigm* (frames, semantic networks, description logics, etc.),
- which *inference mechanisms* are needed (general purpose, automated concept classifier [MacGregor, 1990b], inheritance,[12] monotonic vs modal vs nonmonotonic),
- whether *contexts* are required.

Content features give information about what is represented in the ontology and how that knowledge is represented. One needs to know whether the ontology has an adequate *level of detail*, that is, enough intermediate concepts are represented between two arbitrary concepts. One also needs to know *which concepts are represented in which modules*.

Under the feature *adequacy from the domain expert point of view* several analyses are made: does the content of the ontology include most of the relevant knowledge pieces of the domain? is the terminology adequate? are the definitions adopted correct and widely accepted? is the ontology complete in relation to present needs (at least, one needs to know what important knowledge pieces are missing)? is there superfluous knowledge that should be removed from the ontology while integrating it?

Under the feature *adequacy from the ontologist point of view* several analyses are made: are the basic distinctions represented in the ontology appropriate? does the ontology have an adequate structure? is the ontology structured according to appropriate relations? are needed knowledge pieces represented (this covers issues like "are the appropriate relations represented?", "are certain key concepts represented?")? are those knowledge pieces adequately represented (this covers issues like fidelity, minimal encoding bias, correction, coherence, granularity, conciseness, efficiency in terms of time and space[13])? do they follow adequate naming convention rules? can missing knowledge pieces be added to the ontology without sacrificing coherence and clarity (this covers issues like extendible)? is the ontology clear?

The preponderant parts in this choice are played by the adequacy analyses that domain experts and ontologists have made of the candidate ontologies.

Since this choice is rather complex, simple metrics as the ones proposed to choose candidate ontologies are rather limited. The development of accurate metrics is an important open research area in the OE field.

After the first stage, one has chosen one possible set of ontologies to be integrated. It may be possible to have more than one ontology about one particular domain in that set. Those different ontologies represent knowledge about the same domain from different perspectives. Those different perspec-

[12] Which kind? defeasible, strict, mixed; credulous vs skeptical; on-path vs off-path; bottom-up vs top-down.

[13] It is important to know if we are not reusing an ontology that is not going to meet our needs and the means that we currently have at our disposal.

- content
  - completeness
  - compatibility
    - terminology of common concepts
    - definitions of common concepts

Figure 4: Features for choosing source ontologies, second stage

tives should have been found important to be present in the resulting ontology (there should not be duplicated knowledge represented in the resulting ontology). However, the chosen ontologies may not be compatible among themselves.

Second stage

In the *second stage* one tackles compatibility and completeness of possibly chosen ontologies in relation to the desired resulting ontology, Figure 4.

If the ontologies which are possibly going to be chosen to be integrated are not coherent in what concerns the terminology used and the definitions of the concepts that are common to more than one ontology, then they are not *compatible* and, therefore, cannot be assembled. Sometimes the same concept is named differently in different ontologies. In the resulting ontology one concept only has one denomination, therefore one must be adopted. If one concept has the same definition in all chosen ontologies but different denominations, then a change in terminology can solve the problem. All definitions involving the renamed concept have to be checked and revised accordingly. Sometimes different ontologies adopt different definitions for the same concept. One cannot have this kind of inconsistencies in the resulting ontology. One definition should be chosen and adopted all over. It is more difficult to ensure that the same definition can be adopted by all integrated ontologies. A thorough analysis of all ontologies where one particular concept has a different definition from the adopted one has to be made. It is obvious that only a coherent set of ontologies should be considered for integration purposes.

If chosen ontologies are not *complete*, that is, they do not comprehend all the ontology that has to be built, then this piece of information must be known so that missing knowledge pieces are built from scratch and added or another compatible ontology that contains those knowledge pieces is integrated.

So, although the problem of lack of completeness has to be known, it is not as problematic as lack of coherence. Since one of the issues involved in the domain expert analysis is missing knowledge, one can check whether it is not represented in another ontology about the same domain that is also (or can also be) integrated. However, if chosen ontologies are not compatible among themselves, then this may imply choosing another possible set of ontologies by combining candidate ontologies into a different set, or it may imply building ontologies from scratch (if none of the candidate ontologies adopts the adequate terminology and definitions, or profound changes have to be made to them in order to integrate them).

The problem of choosing the appropriate set of source ontologies is also rather complex. From the set of candidate ontologies, a coherent and adequate subset must be found that is as close as possible to the resulting ontology. Once again, the ontologies in that set may not be perfect candidates. As long as the changes to be made are not very extensive it is more cost effective to reuse the ontology. This analysis has to be performed on a case by case basis. If it is more cost effective to build the ontology from scratch, then existing ontology building methodologies can be used to build an ontology that perfectly suits our needs. If not, ontologies should be reused and integration operations applied so that adequate changes transform the ontologies into perfect candidates.

The result of this activity is a set of ontologies that can and should be assembled together, a description of lacking knowledge that is going to be built from scratch and included in the resulting ontology (since none of the chosen ontologies has it and that knowledge has been identified as essential knowledge that must exist in the resulting ontology) and a description of the changes that should be performed to the integrated ontologies so that they can be perfect candidates and successfully reused (which is the starting point for the application of the integration operations).

4.4 Integration of knowledge

To *integrate knowledge* one needs integration operations and design criteria to guide their application. Sometimes the adaptation of source ontologies may require restructuring activities similar to those that are performed in reengineering processes. Moreover, it may require introduction/removal of knowledge pieces, correction and improvement of the definitions, terminology and documentation of the knowledge pieces represented in the ontology, etc. These adaptations transform the chosen ontology (whole of it) into the needed ontology. In [Farquhar *et al.*, 1997; Borst, 1997; Pinto and Martins, 2000; Pinto, 1999a] initial sets of integration operations are proposed. *Integration operations* can be divided into two groups: basic and non-basic. While the former operations can be algebraically specified the latter can be defined from the former but are custom-tailored operations to be defined in a case by case basis. We have developed an algebraic specification of 39 basic integration operations and specified how 12 non-basic operations can be defined from the previous ones. They are described in [Pinto, 1999b]. We identified a set of *criteria* to guide integration of knowledge: modularize, specialize, diversify each hierarchy, minimize the semantic distance between sibling concepts, maximize relationships between taxonomies and standardize names of relations. They are described in detail in [Arpirez-Vega *et al.*, 1998].

4.5 Analysis of resulting ontology

To *analyze the resulting ontology* one uses a set of features. Besides having an adequate design according to the set of features proposed in [Gruber, 1995][14] and compliance with evaluation criteria [Gómez-Pérez *et al.*, 1995;

[14] Clarity, coherence, extendibility, minimal encoding bias and minimal ontological commitment.

Gómez-Pérez, 1996; 1999][15], one should pay attention to whether the ontology has a *regular level of detail all over*. By regular level of detail we mean that there are no "islands" of exaggerated level of detail and other parts with an adequate one. It should be stressed that none of the parts should have less level of detail than the required one or else the ontology would be useless, since it would not have sufficient knowledge represented. It should also be noted that the other features involved in evaluation and design criteria are analyzed in relation to the resulting ontology, for instance, the resulting ontology should be consistent and coherent all over (although composed by knowledge from different ontologies).

5 Conclusions

In this article we presented the characterization of the ontology integration process. The activities that compose this process are described. The most important activities that form this process include: finding and choosing candidate ontologies, integration oriented evaluation and assessment of candidate ontologies, choosing adequate source ontologies to be integrated, application of integration operations to integrate knowledge and analysis of the resulting ontology. We describe the methods developed to perform these activities. They provide support and guidance to the activities that compose the integration process. They form an integration methodology.

The advantages of the proposed integration methodology are a direct consequence of its generality. One of the advantages of our integration methodology is the fact that it *can be used with different methodologies to build ontologies from scratch*. The only assumption made by this methodology is that knowledge should be represented at the knowledge level.

Special emphasis is given to the *quality* of the ontologies involved in a particular integration process. There are two cases in what regards the ontologies that are reused: (1) they are available at ontology libraries and were built by others or (2) they were built by us. Our methodology proposes that all reused ontologies should be evaluated by domain experts from a technical point of view and assessed by ontologists (more precisely by the ontologists that are going to play the role of integrators) from a user point of view. Integration-oriented technical evaluation and user assessment criteria assure that reused ontologies have enough technical quality to be used in the process. The analysis of the resulting ontology assures that the resulting ontology has enough quality to be made available and (re)used.

References

[Amaya, 1998] M. Dolores Rojas Amaya. Ontologia de Iones Monoatómicos en Variables Fisicas del Medio Ambiente. Proyecto Fin de Carrera, Fac. de Informática, UPM, 1998.

[Arpirez-Vega *et al.*, 1998] J. Arpirez-Vega, A. Gomez-Perez, A. Lozano-Tello, and H. Sofia Pinto. (ONTO)^2 Agent: An Ontology-Based WWW Broker to Select Ontologies. In *Proceedings of ECAI98's Workshop on Application of Ontologies and Problem Solving Methods*, pages 16–24, 1998.

[Arpirez-Vega *et al.*, 2000] J. Arpirez-Vega, A. Gomez-Perez, A. Lozano-Tello, and H. Sofia Pinto. Reference Ontology and (ONTO)^2 Agent: the Ontology Yellow Pages. *Knowledge and Information Systems*, 2(4):387–412, 2000.

[Benjamins and Fensel, 1998] Richard Benjamins and Dieter Fensel. The Ontological Engineering Initiative (KA)^2. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 287–301. IOS Press, 1998.

[Benjamins *et al.*, 1999] Richard Benjamins, Dieter Fensel, Stefan Decker, and Asunción Gómez-Pérez. (KA)^2: Building Ontologies for the Internet, a Mid Term Report. *International Journal of Human Computer Studies*, 51:687–712, 1999.

[Blázquez *et al.*, 1998] M. Blázquez, Mariano Fernández, J. M. García-Pinar, and Asunción Goméz-Pérez. Building Ontologies at the Knowledge Level Using the Ontology Design Environment. In *Proceedings of the Knowledge Acquisition Workshop, KAW98*, 1998.

[Borst, 1997] Pim Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, Tweente University, 1997.

[Chandrasekaran *et al.*, 1999] B. Chandrasekaran, J.R. Josephson, and V. Richard Benjamins. Ontologies: What are they? Why do we need them? *IEEE Expert (Intelligent Systems and Their Applications)*, 14(1):20–26, 1999.

[Farquhar *et al.*, 1996] Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. In *Proceedings of the Knowledge Acquisition Workshop, KAW96*, 1996.

[Farquhar *et al.*, 1997] Adam Farquhar, Richard Fikes, and James Rice. Tools for Assembling Modular Ontologies in Ontolingua. In *AAAI97 Proceedings*, pages 436–441. AAAI Press, 1997.

[Fernández *et al.*, 1997] Mariano Fernández, Asunción Gómez-Pérez, and N. Juristo. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Proceedings of AAAI97 Spring Symposium Series, Workshop on Ontological Engineering*, pages 33–40, 1997.

[Fernández *et al.*, 1999] Mariano Fernández, Asunción Gómez-Pérez, Alexandro Pazos Sierra, and Juan Pazos Sierra. Building a Chemical Ontology Using METHONTOLOGY and the Ontology Design Environment. *IEEE Expert (Intelligent Systems and Their Applications)*, 14(1):37–46, 1999.

[Gangemi *et al.*, 1998] Aldo Gangemi, Domenico M. Pisanelli, and Geri Steve. Ontology Integration: Experiences with Medical Terminologies. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 163–178. IOS Press, 1998.

[15] Correctness –lexically and syntactically–, completeness, conciseness, consistency, expandability, sensitiveness and robustness.

[Gómez-Pérez and Rojas-Amaya, 1999] Asunción Gómez-Pérez and Dolores Rojas-Amaya. Ontological Reengineering for Reuse. In D. Fensel and R. Studer, editors, *Proceedings of the European Knowledge Acquisition Workshop, EKAW99*. Springer Verlag, 1999.

[Gómez-Pérez *et al.*, 1995] A. Gómez-Pérez, N. Juristo, and J. Pazos. Evaluation and Assessment of the Knowledge Sharing Technology. In N.J.I. Mars, editor, *Towards Very Large Knowledge Bases*, pages 289–296. IOS Press, 1995.

[Gómez-Pérez *et al.*, 1996] Asunción Gómez-Pérez, Mariano Fernández, and António J. de Vicente. Towards a Method to Conceptualize Domain Ontologies. In *Proceedings of ECAI96's Workshop on Ontological Engineering*, pages 41–52, 1996.

[Gómez-Pérez, 1996] Asunción Gómez-Pérez. Towards a Framework to Verify Knowledge Sharing Technology. *Expert Systems with Applications*, 11(4):519–529, 1996.

[Gómez-Pérez, 1999] Asunción Gómez-Pérez. Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases. In *Proceedings of the Knowledge Acquisition Workshop, KAW99*, 1999.

[Gruber and Olsen, 1994] Thomas Gruber and G. R. Olsen. An Ontology for Engineering Mathematics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *KR94 Proceedings*, pages 258–269. Morgan Kaufmann, 1994.

[Gruber, 1995] Thomas Gruber. Towards Principles for the Design of Ontologies for Knowledge Sharing. *International Journal of Human Computer Studies*, 43(5/6):907–928, 1995.

[Gruninger, 1996] Michael Gruninger. Designing and Evaluating Generic Ontologies. In *Proceedings of ECAI96's Workshop on Ontological Engineering*, pages 53–64, 1996.

[MacGregor, 1990a] Robert MacGregor. LOOM User Manual. Technical Report ISI/WP-22, USC/Information Sciences Institute, 1990.

[MacGregor, 1990b] Robert MacGregor. The Evolving Technology of Classification-Based Representation Systems. In John Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 385–400. Morgan Kaufman, 1990.

[McGuiness *et al.*, 2000] Deborah L. McGuiness, Richard Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In Anthony Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000 Proceedings*, pages 483–493. Morgan Kaufmann, 2000.

[Newell, 1982] A. Newell. The Knowledge Level. *Artificial Intelligence*, 18(1):87–127, 1982.

[Noy and Musen, 1999] Natalya Fridman Noy and Mark A. Musen. An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support. In *Proceedings of AAAI99's Workshop on Ontology Management, WS-99-13*, pages 17–27. AAAI Press, 1999.

[Noy and Musen, 2000] Natalya Fridman Noy and Mark A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI2000 Proceedings*, pages 450–455. AAAI Press, 2000.

[Pinto and Martins, 2000] H. Sofia Pinto and J.P. Martins. Reusing Ontologies. In *Proceedings of AAAI 2000 Spring Symposium Series, Workshop on Bringing Knowledge to Business Processes, SS-00-03*, pages 77–84. AAAI Press, 2000.

[Pinto *et al.*, 1999] H. Sofia Pinto, A. Gómez-Pérez, and J. P. Martins. Some Issues on Ontology Integration. In *Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, pages 7.1–7.12, 1999.

[Pinto, 1999a] H. Sofia Pinto. Towards Ontology Reuse. In *Proceedings of AAAI99's Workshop on Ontology Management, WS-99-13*, pages 67–73. AAAI Press, 1999.

[Pinto, 1999b] H. Sofia Pinto. Towards operations to ontology integration. Technical Report GIA 99/02, Grupo de Inteligência Artificial do Instituto Superior Técnico, April 1999.

[Russ *et al.*, 1999] Thomas Russ, Andre Valente, Robert MacGregor, and William Swartout. Practical Experiences in Trading Off Ontology Usability and Reusability. In *Proceedings of the Knowledge Acquisition Workshop, KAW99*, 1999.

[Sowa, 2000] John Sowa. *Knowledge Representation: logical, philosophical and computational foundations*. Brooks/Cole, 2000.

[Swartout *et al.*, 1997] Bill Swartout, Ramesh Patil, Kevin Knight, and Tom Russ. Toward Distributed Use of Large-Scale Ontologies. In *Proceedings of AAAI97 Spring Symposium Series, Workshop on Ontological Engineering*, pages 138–148, 1997.

[Uschold and Gruninger, 1996] Mike Uschold and Michael Gruninger. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11(2), June 1996.

[Uschold and King, 1995] Mike Uschold and Martin King. Towards a Methodology for Building Ontologies. In *Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[Uschold *et al.*, 1998] Mike Uschold, Mike Healy, Keith Williamson, Peter Clark, and Steven Woods. Ontology Reuse and Application. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 179–192. IOS Press, 1998.

[Wiederhold, 1994] Gio Wiederhold. Interoperation, Mediation and Ontologies. In *Proceedings of the International Symposium on the Fifth Generation Computer Systems, Workshop on Heterogeneous Cooperative Knowledge-Bases*, volume W3, pages 33–48, 1994.

Building a Reason-able Bioinformatics Ontology Using OIL

Robert Stevens, Ian Horrocks, Carole Goble and Sean Bechhofer
Department of Computer Science
University of Manchester
Oxford Road
Manchester, M13 9PL
United Kingdom
robert.stevens@cs.man.ac.uk

Abstract

Ontologies will play an important role in bioinformatics, as they do in other disciplines, where they will provide a source of precisely defined terms that can be communicated across people and applications.

The Ontology Inference Layer (OIL), is an ontology language that has an easy to use frame feel, yet at the same time allows users to exploit the full power of an expressive description logic. OilEd, an editor for OIL, uses reasoning to support ontology design, facilitating the development of ontologies that are both more detailed and more accurate.

This paper presents a bioinformatics ontology building case study using OilEd to highlight the features of the combination of a frame representation and an expressive description logic.

1 Introduction

Ontologies have become an increasingly important research topic. This is chiefly a result of their usefulness in a range of application domains [van Heijst *et al.*, 1997; McGuinness, 1998; Uschold and Grüninger, 1996] including bioinformatics [Stevens *et al.*, 2001].

Biologists have long had a culture of recording and sharing information. This information has traditionally been stored in natural language form and latterly in natural language annotated databases. There has been a recognition that if these information resources are to continue to play their central role in bioinformatics, they have to become machine understandable.

The automation of tasks depends on elevating the status of the information in resources from machine-readable to something we might call machine-understandable. The natural language annotation of a bioinformatics resource can be processed computationally, but using the knowledge contained in the natural language annotation is difficult. The key idea is to have data in these resources defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans.

To realise this goal, it will be necessary to annotate bioinformatics resources with *metadata* (i.e., data describing their content/functionality). Ontologies are a useful mechanism to provide metadata for various resources. However, such annotations will be of limited value to automated processes unless they share a common understanding as to their meaning. Ontologies, can help to meet this requirement by providing a "representation of a shared conceptualisation of a particular domain" that can be communicated across people and applications [Gruber, 1993].

There have been several attempts to develop bioinformatics ontologies to exploit this biological information. The Gene Ontology (GO) [The Gene Ontology Consortium, 2000] is a controlled vocabulary for annotating gene products for molecular functions, the biological processes in which it is involved and the cellular locations in which it is found. EcoCyc [Karp *et al.*, 2000] has used an ontology to specify a database schema for the E. *coli*. metabolism, signal trandsduction etc. RiboWeb [Altman *et al.*, 1999] also uses an ontology to describe its data, but also guide its users through analysis of their data. Finally, TAMBIS (Transparent Access to Multiple bioinformatics Information Sources) [Baker *et al.*, 1998] uses an ontology to allow users to query bioinformatics databases. Each of these uses a different knowledge representation system from phrases in GO; to frame based systems in EcoCyc and RiboWeb to a description logic in TAMBIS.

Phrase based vocabularies have the advantage of being easily accessible, but suffer from difficulties in consistency and maintenance. It is common for mistakes to be made in phrase based vocabularies, especially in the maintenance of multiple hierarchies. Frame-based systems have the advantage of an easily accessible and intuitive modelling style, reminisient of an object view of the world (a frame is a class and the slots are attributes. The frame encapsulates the properties of the instances). Such systems, like phrase based vocabularies, are esentially hand-crafted and can suffer from inconsistencies and logical mistakes. The well defined semantics and reasoning support of DLs allow logically consistent ontologies to be maintained. Concepts can be defined in terms of their properties and the reasoning used to classify the concepts based upon those descriptions. When an concept expression is unsatisfyable in terms of the rest of the model, the reasoning support can inform the modeller of his or her mistake. Description logic based ontologies avoid the problems of the hand-crafted ontologies, but suffer from the complexity of the modelling style.

These considerations have led to the development of OIL [Fensel *et al.*, 2000], an ontology language that extends a frame-based like view with a much richer set of modelling primitives[1]. OIL has a frame-like syntax, which facilitates tool building, yet can be mapped onto an expressive description logic (DL), which facilitates the provision of reasoning services. Thus a modeller is offered the best of both worlds in both development and deployment of an ontology. OilEd is an ontology editing tool for OIL (and DAML+OIL) that exploits both these features in order to provide a familiar and intuitive style of user interface with the added benefit of reasoning support. Its main novelty lies in the extension of the frame editor paradigm to deal with a very expressive language, and the use of a highly optimised DL reasoning engine to provide sound and complete, yet still empirically tractable reasoning services.

Reasoning with terms from deployed ontologies will be valuable in many bioinformatics applications. The most obvious is in formulating, processing and answering queries over bioinformatics databases. There is also a great potential in using reasoning during analysis of novel biological entities.

The reasoning support offered by OIL is also extremely valuable at the ontology design phase, where it can be used to detect logically inconsistent classes and to discover implicit subclass relations. This encourages a more descriptive approach to ontology design, with the reasoner being used to infer part of the subsumption lattice (see the case study presented in Section 4); the resulting ontologies contain fewer errors of consistency, yet provide more detailed descriptions that can be exploited by automated processes in the deployed ontologies. Finally, reasoning is of particular benefit when ontologies are large and/or multiply authored, and also facilitates ontology sharing, merging and integration [McGuinness *et al.*, 2000]; considerations that will be particularly important in the distributed bioinformatics environment.

The modeller, however, is not forced to use reasoning support OilEd can be used to construct hierarchies of terms unadorned by descriptions of properties. In fact, the ontology development described in Section 4, uses a cyclic, two stage approach to ontology development. For a given portion of the ontology, a simple hierarchy of terms is hand-crafted. In the next stage, properties are described for the necessary and sufficient conditions to be a member of that class or concept. The reasoner can then be used to check the descriptions for logical consistency and offer any inferred knowledge (unknown subsumptions) that have been found. The cycle is then repeated for this and other portions of the ontology.

This paper concentrates on this ontology design use of OIL, rather than the use of reasoning in the deployed ontologies. A case study taken from the domain of bioinformatics will be used to highlight the development and management facilities

afforded by the combination of the frame-like syntax of OIL with the expressive power of description logics. First, the

principal features of the language (Section 2) and its associated editor

(Section 3) will be described. Section 4 describes the development of an ontology of molecular biology and bioinformatics using OilEd.

2 Oil and DAML+OIL

The development of OIL resulted from efforts to combine the best features of frame and DL based knowledge representation systems, while at the same time maximising compatibility with emerging web standards. These standards, such as RDFS [Brickley and Guha, 2000], make it easier to use ontologies consistently across the web. The intention was to design a language that was intuitive to human users, and yet provided adequate expressive power for realistic applications (many early DLs failed on this second count—see [Doyle and Patil, 1991]).

The resulting language combines a familiar frame like syntax (derived in part from the OKBC-lite knowledge model [Chaudhri *et al.*, 1998]), with the power and flexibility of a DL (i.e., boolean connectives, unlimited nesting of class elements, transitive and inverse slots, general axioms, etc.). The language is defined as an extension of RDFS, thereby making OIL ontologies (partially) accessible to any "RDFS-aware" application.

The frame syntax is less daunting to ontologists/domain experts than a DL style syntax, and it facilitates a modelling style in which ontologies can start out simple (in terms of their descriptive content) and are gradually extended, both as the design itself is refined and as users become more familiar with the language's advanced features (see Section 4). The frame paradigm also facilitates the construction and adaptation of tools, e.g., the OntoEdit and Protégé editors and the Chimaera integration tool are all being adapted to use OIL/DAML+OIL [Staab and Maedche, 2000; Grosso *et al.*, 1999; McGuinness *et al.*, 2000].

On the other hand, basing the language on an underlying mapping to a very expressive DL ($\mathcal{SHIQ}$) provides a well defined semantics and a clear understanding of its formal properties, in particular that the class subsumption/satisfiability problem is decidable and has worst case ExpTime complexity [Horrocks *et al.*, 1999]. The mapping also provides a mechanism for the provision of practical reasoning services by exploiting implemented DL systems, e.g., the FaCT system [Horrocks, 2000].

OIL extends standard frame languages in a number of directions. One of the key ideas is that an anonymous class description, or even boolean combinations of class descriptions, can occur anywhere that a class name would ordinarily be used, e.g., in slot constraints and in the list of superclasses. For example, in Figure 1 (which uses OIL's "human readable" presentation syntax, rather than the more verbose RDFS serialisation), a herbivore is described as an animal that eats only plants or part-of plants. Points to note are that universally quantified (value-type) and existentially quantified (has-value) slot constraints are clearly differentiated, and that the constraint on the eats slot is a disjunction,

[1] A similar ontology language called DAML has been developed as part of the DARPA DAML project [Hendler and McGuinness, 2001]. These two languages are soon to be merged under the name DAML+OIL.

one of whose components is an anonymous class description (in this case, just a single slot constraint). In addition, it is asserted that the part-of slot is transitive, and that its inverse is the slot has-part. Further details of the language will be given in Section 3, and a complete specification can be found in [Fensel *et al.*, 2000].

slot-def part-of
  subslot-of structural-relation
  inverse has-part
  properties transitive
class-def defined herbivore
  subclass-of animal
  slot-constraint eats
    value-type plant *or*
      slot-constraint part-of *has-value* plant

Figure 1: OIL language example

3 OilEd

OilEd is a simple ontology editor that supports the construction of OIL-based ontologies. The basic design has been heavily influenced by similar tools such as Protégé [Grosso *et al.*, 1999] and OntoEdit [Staab and Maedche, 2000], but OilEd extends these approaches in a number of ways, notably through an extension of expressive power and the use of a reasoner.

3.1 OilEd Functionality

Basic functionality allows the definition and description of classes, slots, individuals and axioms within an ontology.

In general, editing functions are provided through graphical means—mouse driven drop down menus, toolbars and buttons. We will not provide a detailed description of the graphical user interface here, as it is relatively standard (see Figure 2, which provides a screen shot of the editors class definition panel). Instead, we will discuss the novel functionality offered by the tool.

Frame Descriptions The central component used throughout OilEd is the notion of a *frame description*. This consists of a collection of superclasses along with a list of slot constraints. For example, a class called hydrolase has constraints including catalyses hydrolysis, describing one of the properties of a hydrolase to be the promotion of the reaction called hydrolysis. This is similar to other frame systems. Where OilEd differs, however, is that wherever a class name can appear, a recursively defined, anonymous frame description can be used. For example, a gene is has-name gene-name or part-of gene-name – indicating that a gene may be found using its name or part of its name. In addition, arbitrary boolean combinations of frames or classes (using **and**, **or** and **not**) can also appear. This is in contrast to conventional frame systems, where in general, slot constraints and superclasses must be class names.

As well as being able to assert individuals as slot fillers, several types of constraints on slot fillers can be asserted

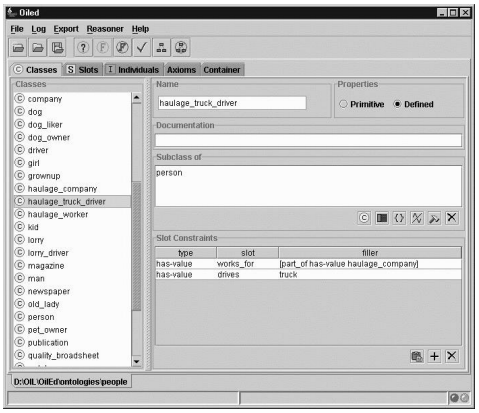


Figure 2: OilEd Class Panel

(these kinds of constraint are sometimes called *facets*). These include *value-type* restrictions (all fillers must be of a particular class), *has-value* restrictions (there must be at least one filler of a particular class), and explicit *cardinality* restrictions (e.g., at most three fillers of a given class). For instance, it is possible to exactly describe that a G-protein coupled receptor has to have seven and only seven transmembrane regions – otherwise it is not a G-protein coupled receptor. Each constraint has a clearly defined meaning, removing the confusion present in some frame systems, where, for example, it is not always clear whether the semantics of a slot-constraint should be interpreted as a universal or existential quantification.

Class Definitions A class definition specifies the class name, along with an optional frame description (see above) and a specification of whether the class is *defined* or *primitive*. If defined, the class is taken to be equivalent to the given description (necessary and sufficient conditions). If primitive, the class is taken to be an explicit subclass of the given description (necessary conditions). In the specification of the OIL language, classes can have multiple definitions. In OilEd, this is not allowed—classes must have a single definition—but the same effect can be achieved through the use of *equivalence* axioms as discussed below.

Slot Definitions A slot definition gives the name of the slot and allows additional properties of the slot to be asserted, e.g., the names of any *superslots* or *inverses*. If r is a superslot of s, then any two objects related via s must also be related via r (i.e., $\mathsf{s}(a,b) \rightarrow \mathsf{r}(a,b)$); if r is an inverse of s, then a is related to b via s iff b is related to a via r (i.e., $\mathsf{s}(a,b) \leftrightarrow \mathsf{r}(b,a)$). Domain and range restrictions on a slot can also be specified. For example, we can constrain the relationship parent to have both domain and range person, asserting that only persons can have, and be, parents. As with class descriptions, the domain and range restrictions can be arbitrary class expressions such as anonymous frames or boolean combinations of classes or frames, again extending the expressivity of traditional frame editors. Note that in this context, the domain and range restrictions are *global*, and apply to every occurrence of the slot, whether explicit or implicit.

A slot r can also be asserted to be transitive (i.e., $\mathsf{r}(a,b)$ and

$\mathsf{r}(b, c) \rightarrow \mathsf{r}(a, c)$), functional (i.e., $\mathsf{r}(a, b)$ and $\mathsf{r}(a, c) \rightarrow b = c$) or symmetric (i.e., $\mathsf{r}(a, b) \rightarrow \mathsf{r}(b, a)$).

All assertions made about slots are used by the reasoner, and may induce hierarchical relationships between classes, e.g., as a result of domain and range restrictions (see Section 4).

Axioms Another area where the expressive power of OIL/OilEd exceeds that of traditional frame languages/editors is in the kinds of *axiom* that can be used to assert facts about classes and their relationships. As well as standard class definitions (which are really a restricted form of subsumption/equivalence axiom), OilEd axioms can also be used to assert the *disjointness* or *equivalence* of classes (with the expected semantics) along with *coverings*. A covering asserts that every instance of the covered class must also be an instance of at least one of the covering classes. In addition, coverings can be said to be *disjoint*, in which case every instance of the covered class must be an instance of exactly one of the covering classes.

Again, these axioms are not restricted to class names, but can involve arbitrary class expressions (anonymous frames or boolean combinations). This is a very powerful feature, and is one of the main reasons for the high complexity of the underlying decision problem. These axioms, especially the disjointness axiom, are quite heavily used in the case study ontology. It is useful to state explicitly that, for instance, something cannot be both an element and a compound.

Individuals Limited functionality is provided to support the introduction and description of individuals—the intention within OilEd is that such individuals are for use within class descriptions, rather than supporting the production of large existential knowledge bases (it is supposed that RDF/RDFS will be used directly for this purpose). As a (non-biological) example, we may wish to define the class of Italians as being all those Persons who were born in Italy, where Italy is not a class but an individual. The example ontology in Section 4 does not use any individuals. It might, however, be possible to use them to describe individual chemicals within the ontology.

Concrete Datatypes Concrete datatypes (string and integers), along with expressions concerning concrete datatypes (such as min, max or ranges) can also be used within class descriptions. However, the FaCT reasoner does not support reasoning over concrete datatypes, and at present OilEd simply ignores concrete datatype restrictions when reasoning about ontologies. The theory underlying concrete datatypes is, however, well understood [Baader and Hanschke, 1991], and work is in progress to extend the FaCT reasoner with support for concrete datatypes. These data types are used in the description of atom in the example ontology (Section 4).

3.2 Reasoning

The editor can be requested to verify an ontology using the FaCT reasoner. When verification is requested, the ontology is translated into an equivalent $\mathcal{SHIQ}$ (or $\mathcal{SHF}$) knowledge base and sent to the reasoner for classification [Decker *et al.*, 2000]. OilEd then queries the classified knowledge base, checking for inconsistent classes and implicit subsumption relationships. The results are reported to the user by highlighting inconsistent classes and rearranging the class hierarchy display to reflect any changes discovered. FaCT/OilEd does not provide any explanation of its inferences, although this would clearly be useful in ontology design [McGuinness and Borgida, 1995].

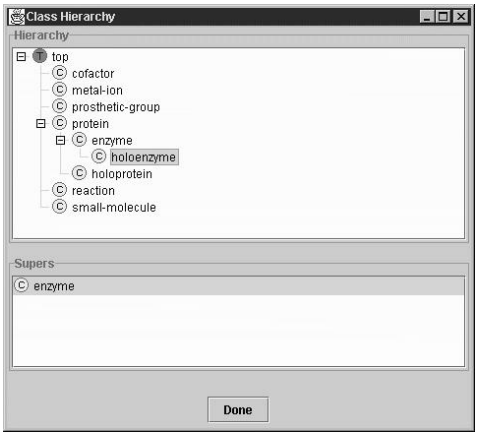


Figure 3: Hierarchy pre-classification

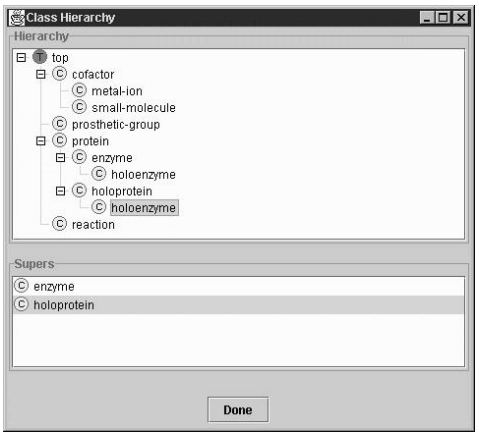


Figure 4: Hierarchy post-classification

Figures 3 and 4 show the effects of classification on (part of) the hierarchy derived from the TAMBIS ontology (see Section 4). When verifying the ontology, a number of new subsumption relationships are discovered (due to the class definitions in the model).

In particular we can see that, after verification, holoenzyme is not only an enzyme, but also a holoprotein, and that metal-ion and small-molecule are both subclasses of cofactor. Note that if the reasoning is not employed, and if the extended expressiveness and advanced features are not used, OilEd will still function as a simple frame editor.

4 Case Study: the TAMBIS Ontology

The role of ontologies in bioinformatics has become prominent in the last few years. Much of biology works by applying prior knowledge to an unknown entity. The complex biolog-

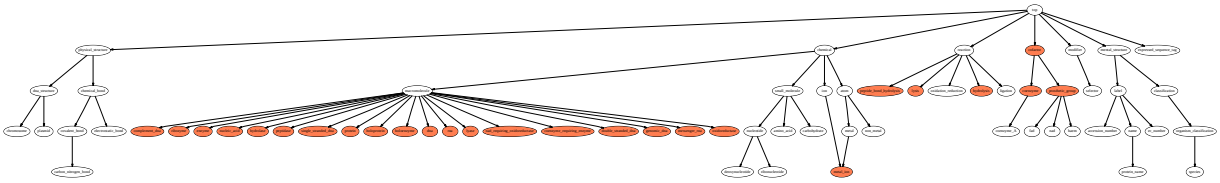


Figure 5: Definitions pre-classification

ical data stored in bioinformatics databases requires knowledge to specify and constrain values held in that database. Ontologies are also used as a mechanism for expressing and sharing community knowledge, to define common vocabularies (e.g., for database annotations), and to support intelligent querying over multiple databases [Baker *et al.*, 1999; Stevens *et al.*, 2001].

TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) is a mediation system that uses an ontology to enable biologists to ask questions over multiple external databases using a common query interface. The ontology is central to the TAMBIS system: it provides a model over which queries can be formed, it drives the query formulation interface, it indexes the middleware wrappers of the component sources, and it supports the query rewriting process [Goble *et al.*, 2001]. The TAMBIS ontology (TaO) covers the principal concepts of molecular biology and bioinformatics: macromolecules; their motifs, their structure, function, cellular location and the processes in which they act. It is an ontology intended for retrieval purposes rather than hypothesis generation, so it is broad and shallow rather than deep and narrow [Baker *et al.*, 1999].

The TaO was originally modelled in the GRAIL DL [Rector *et al.*, 1997]. It was subsequently migrated to OIL in order to (a) exploit OIL's high expressivity, maintaining a better fidelity with biological knowledge as it is currently perceived; (b) use reasoning support when building and evolving complex ontologies where the knowledge is dynamic and shifting; and (c) be able to deliver the TaO as a conventional frame ontology (with all subsumptions made explicit), thus making it accessible to a wider range of (legacy) applications and collaborators.

The approach to developing the ontology was directly influenced by the range of expressivity that OIL affords, and the capabilities of OilEd itself, particularly its reasoning facilities. The modelling philosophy was to be descriptive, i.e., to model properties and allow as much as possible of the subsumption lattice to be inferred by the reasoner.

The design methodology was to first construct a basic framework of primitive foundation classes and slots, working both top down and bottom up, mainly using explicitly stated superclasses. This was a cyclic activity, with portions of the TaO being described primitively, then in the more descriptive fashion.

In each cycle, the reasoner is used to classify the ontology. The classification can then be viewed (with and without inferred subsumptions) to check the classification against the ontologist's knowledge. The editor allows concepts to be found by name, so recently constructed concepts can be viewed in their context. Logically inconsistent concept expressions (those equivalent to bottom) are higlighted for easy identification of badly formed expressions.

The initial model was very "tree-like", i.e., there were very few classes with multiple superclasses. The primitive portions of the ontology were then incrementally extended and refined by adding new classes, elaborating slot fillers and constraints, and "upgrading" to defined classes wherever possible, so that class specifications became steadily more detailed and faithful to the application. This process was guided by subsumption reasoning—when elaborating or changing classes, the reasoner could be used to check consistency and to show the impact on the class hierarchy.

As each cycle of extension of concept definitions ends, the modeller is able to view the use of primitive and defined concepts across the ontology. This view 'zooms' out from the ontology, showing the lattice as dots and arcs, with the dots differentiated according to their being defined or primitive. This enables the modeller to see areas of definition and where definition is lacking. Building-block concepts, that are not central to the use of the ontology, will in all likelyhood remain primitive, but it is useful to spot where definition is lacking; as definition increases the fidelity and justification for the ontology. For instance, the macromolecules within the TaO are highly defined, but the properties, used in the definition of more central concepts remain primitive.

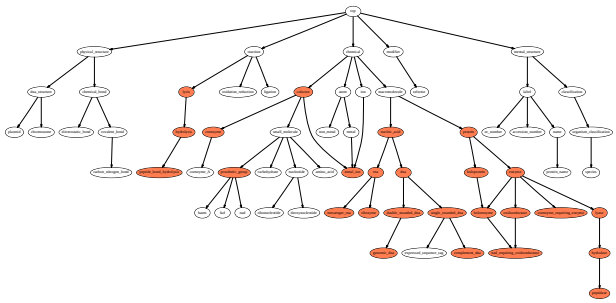


Figure 6: Definitions post-classification

Figures 5 and 6 illustrate this (using a subset of the complete ontology). Figure 5 shows the distrubution of defined concepts throughout the hierarchy before classification[2]. Defined concepts are signified using a darker colour, and we can see that the hierarchy has a very flat structure. In Figure 6,

[2]The hierarchies are generated using OilEd's export functionality, which produces graphs for rendering by AT&T's Graphviz software

we see the situation after classification. The defined concepts have now been organised into a subsumption hierarchy based on their definitions.

Figure 7 shows a (greatly simplified) fragment of the TaO (using OIL's presentation syntax) that we will use to illustrate this methodology.[3]

class-def protein

class-def defined holoprotein
  subclass-of protein
  slot-constraint binds
    has-value prosthetic-group

class-def defined enzyme
  subclass-of protein
  slot-constraint catalyses
    has-value reaction

class-def defined holoenzyme
  subclass-of enzyme
    slot-constraint binds *has-value* prosthetic-group

class-def defined cofactor
  subclass-of (metal-ion *or* small-molecule)

disjoint metal-ion small-molecule

Figure 7: Simplified fragment of TAMBIS ontology

Bioinformatics is the study and analysis of molecular biology – the functions and processes of the products of an organism's genes. The knowledge about molecular biology is contained within numerous data banks and analysis tools. An ontology of bioinformatics therefore needs to support two domains: First, the domain of molecular biology – the chemicals and higher-order chemical structures within a cell and second, to reflect the nature and content of bioinformatics resources.

The TaO was built with both a top-down and bottom-up strategy. A general domain framework was provided into which more detailed molecular biological and bioinformatics concepts could be fitted. As well as this approach, a solid conceptual foundation about chemicals and their structure and behaviour was built. Basic chemicals and their properties are used to describe the more complex biological molecules of interest to bioinformatics, so this is an appropriate approach both from a straightforward content, as well as a modelling, point of view. This involved the description of the different kinds of chemicals (ions, atoms and molecules etc.); their structure, reactions, function and processes in which they act. This general foundation was then used to give the subsequent detailed description of the salient molecular biological concepts that form the bottom-up placement of defined concepts.

The core of the TaO is a description of basic chemical concepts. The various kinds of chemical are defined as children of the concept chemical. These include:

atom The building block of all chemicals. A chemical's behaviour is defined by the number of protons it contains, i.e., its atomic number. Therefore, atom is defined as:

[3]The complete ontology can be found at `http://img.cs.man.ac.uk/stevens/tambis-oil.html`

class-def defined atom
  subclass-of chemical
  slot-constraint atomic-number
    cardinality 1
    value-type integer
    has-value (*min* 1)

So, atoms may only have one atomic number, which must be an integer greater than or equal to 1. The concepts metal-atom, nonmetal-atom and metalloid-atom are defined to be atoms with the physicochemicalproperty of either metal nonmetal or metalloid respectively.

The concept of carbon has been defined as a kind of atom with atomic number six and the physicochemical property of non-metal. This description of the concept carbon enables it to be automatically placed as a kind of nonmetal-atom. Several other, biologically relevant, atom types have been included in the TaO.

ion An ion is simply a chemical with an electrical charge. It is defined as:

class-def defined ion
  subclass-of chemical
  slot-constraint has-charge
    has-value (*not* 0)

The slot constraint describes that an ion must have an electrical charge and it can only be an electrical charge. It also describes that the value for this charge can be a positive or negative number, but not zero. it would be possible to capture chemical reality further by specifying a minimum cardinality of one – that is, a chemical must have at least one charge to be an ion, but may have more than one charge (a molecule could, for instance, contain both a positive and negative charge).

the chemical ion has two asserted children: cation and anion. Defining cation as a chemical with charge greater-than 0 enables the classifier to place it correctly as a kind of ion. An equivalence axiom can be used to state that cation is a synonym of positive-ion.

Now,divalent-cation (a chemical with two positive charges) can be defined by adding further properties to this slot constraint: That

the filler for has-charge is equal 2, that is, has positive two charges on the chemical.

element An element is a kind of chemical containing only one kind of atom. OIL has the expressive power to constrain the slot atom-type to be equal to only one. Adding the slot constraint atom-type with the value one to atom would also classify atom as an element.

compound A compound is a chemical containing more than one kind of atom. The slot constraint used for element (above) is altered so that the constraint indicates that at least two kinds of atom must be present in this kind of chemical.

molecule A molecule is a kind of chemical containing atoms linked by covalent bonds. The concept covalent-bond was described as a kind of chemical-structure and used to fill the slot contains-bond, with the *has-value* restriction. So, there must be a covalent bond present for it to be classed as a molecule, but other kinds of bond may be present – exactly capturing what we understand of basic chemicals.

Two principal features of the ontology development arise from this chemical core:

1. The need for a framework of primitive concepts, such as metal and properties such as has-charge. These can be used to develop the core of defined concepts at the centre of the TaO. Primitive concepts, as well as those such as chemical itself, are placed within a simple upper level ontology containing physical, mental, substance, structure, function and process. These are extended by their obvious conjunctive forms, e.g., physical-structure.
2. The ability to rapidly extend this chemicals core to another layer of defined chemical concepts, all of which used the previously defined concepts.

The next "layer" of chemical descriptions included:

molecular-compound A chemical containing covalent bonds and more than one type of atom.

elemental-molecule A chemical, such as oxygen (O_2), that contains covalent bonds and only one kind of atom.

metal-ion A kind of atom with an electrical charge.

ionic-compound A kind of chemical containing more than one kind of atom and has an electrical charge.

All these and more were simply defined to be the conjunction of two concepts. For example:

class-def defined metal-ion
  subclass-of metal, ion

class-def defined divalent-cation
  subclass-of chemical
  slot-constraint has-charge
    has-value (*equal* 2)

A concept divalent-zinc-cation can then simply be defined as:

class-def defined divalent-zinc-cation
  subclass-of zinc
    slot-constraint has-charge
      has-value (*equal* 2)

These descriptions of chemicals can be reinforced with the use of axioms. It is not possible to be both an element and a compound, so these two concepts are described as disjoint. This means that if a concept were to be defined with properties of both an element and a compound, it would be found to be inconsistent by the reasoner. Such strict definitions help maintain the consistency and biological thoroughness of the ontology.

An organic-molecular-compound is a molecular compound that contains at least one carbon atom. This, however, is not sufficient to define an organic molecular compound. Carbon dioxide (CO_2) is a molecular compound containing carbon, but is not organic. Thus the property of containing carbon is only a necessary condition for being an organic molecular compound. Again, the ability to be exact with concept descriptions allows the ontology to match chemical and biological knowledge closely and prevent conceptualisations being made that contradict domain knowledge.

Bioinformatics is mainly concerned with organic macromolecular-compounds. Thus, organic molecular compound was split into the biologically useful distinctions of macromolecular-compound and small-molecular-compound. the distinction is one of size and a protein, for example, of over 100 Daltons is usually said to be a macromolecule. Unfortunately the boundary is more complex, a smaller molecule can still be "macro", depending on its context. For this reason, sufficiency conditions were not used in the definition. Useful small organic molecules were simply asserted as primitive concepts underneath small-organic-molecular-compound. These include nucleotide, amino-acid and others useful in describing the properties of biological concepts.

For the purposes of the TaO, macromolecular-compounds are polymers of small-organic-molecular-compounds and are defined as such. Thus, protein is defined as a polymer of amino-acid; nucleic-acid as a polymer of nucleotide and polysaccaride as a polymer of saccaride. A macromolecule can only be a polymer of one kind of small molecule, so the *value-type* restriction is used in the slot constraint. It is only possible to be one of these molecules, so the *disjoint* axiom is used on these macromolecules.

As most of bioinformatics concentrates on the analysis and description of nucleic acids and proteins, much of the TaO's description concentrates in this area. DNA and RNA are both nucleic acids formed from different kinds of nucleotide.

Describing DNA *slot-constraint value-type has-value* deoxy-nucleotide, allows the classifier to correctly place it as a kind of nucleic-acid and capture that DNA can only be a polymer of the deoxy- form of a nucleotide and some of the nucleotide have to be present. The various different kinds of DNA and RNA are distinguished by their function and/or cellular location. Again, as before, other parts of the TaO are used to describe these properties of biological concepts. For example, genomic-dna is dna that is found on a nuclear chromosome, chloroplast chromosome, or mitochondrial chromosome. The slot constraint uses *or* in the filler class expression to describe this:

slot-constraint part-of
  cardinality 1
  value-type
    (nuclear-chromosome *or*
    mitochondrial-chromosome *or*
    chloroplast-chromosome)).

The TaO contains a rich partonomy. The cellular structures, in particular, use the part-of slot and its transitive property to build up this partonomy. For instance, nuclear-chromosome is part-of the nucleus, which itself is part-of the cell. Thus, a nuclear-chromosome is part-of the cell.

These biological-structures and associated partonomy are part of the TaO. Not only are they used in building some of the descriptions of bio-concepts, but are also part of the description of the content of bioinformatics resources.

In the initial description of kinds of protein, holoprotein, enzyme and holoenzyme were originally primitive classes, with no slot constraints, and an explicitly asserted class hierarchy: holoprotein and enzyme were subclasses of protein, and holoenzyme was a subclass of enzyme.

During the extension and refinement phase, the properties of the various classes were described in more detail: it was asserted that a holoprotein binds a prosthetic-group, that an enzyme catalyses a reaction, and that a holoenzyme binds a prosthetic-group. Several of the classes were also upgraded to being *defined* when their description constituted both necessary and sufficient conditions for class membership, e.g., a protein is a holoprotein if and only if it binds a prosthetic-group.

Enzyme was removed from the superclass list and replaced with protein; then holoenzyme's properties were described in more detail using slot constraints—in particular, it was asserted that a holoenzyme catalyses a reaction and binds a prosthetic-group. This allows the reasoner to infer not only the subclass relationship w.r.t. enzyme, but also additional subclass relationships w.r.t. holoprotein, and in particular that holoenzyme is a subclass of holoprotein. This latter relationship could have been missed if the ontology had been hand crafted.

The extension and refinement phase also included the addition of axioms asserting disjointness, equality and covering, further enhancing the accuracy of the model. Referring again to Figure 7, our biologist initially asserted that cofactor was a subclass of both metal-ion and small-molecule (a common confusion over the semantics of 'and' and 'or') rather than being either a metal-ion or a small-molecule. Subsequently, when it was asserted that metal-ion and small-molecule are disjoint, the reasoner inferred that cofactor was logically inconsistent, and the mistake was rectified. Modelling mistakes such as these litter bioontologies crafted by hand.

There are two kinds of cofactor – coenzyme and prosthetic-group. A coenzyme can be either a small molecule or metal ion and binds loosely to a protein. A prosthetic group, on the other hand, is a kind of cofactor that binds tightly to a protein, but can only be a small molecule. Again, OIL is expressive enough to capture these distinctions accurately.

class-def defined prosthetic-group
  subclass-of cofactor *and* (*not* metal-ion)
  slot-constraint binds-tightly
    has-value protein

The slot hierarchy was also used to induce the classification of types of enzyme. For example, reaction (used in the definition of enzyme) has a child lysis. Lysis is the breaking of a covalent bond and hydrolysis is breaking of a covalent bond with water. These two reactions are defined using the following slot definitions:

slot-def lysis-of
  domain reaction
  range covalent-bond

slot-def hydrolysis-of
  subslot-of lysis-of

class-def defined lysis
  subclass-of reaction
  slot-constraint lysis-of
    has-value covalent-bond
    value-type covalent-bond

class-def defined hydrolysis
  subclass-of reaction
  slot-constraint hydrolysis-of
    has-value covalent-bond
    value-type covalent-bond

class-def defined lyase
  subclass-of protein
  slot-constraint catalyses
    has-value lysis
    value-type lysis

class-def defined hydrolase
  subclass-of protein
  slot-constraint catalyses
    has-value hydrolysis
    value-type hydrolysis

A lyase is a protein that catalyses lysis. A hydrolase is a protein that catalyses hydrolysis. As the slot hierarchy describes hydrolysis-of being a subslot of lysis-of, hydrolysis is a child of lysis and consequently, hydrolase is a child of lyase.

Other advantages derived from the use of OilEd included:

- The frame-like look and feel of OilEd, and the frame approach of the OIL language, made ontology development much less daunting to our biologist than writing $\mathcal{SHIQ}$ logic expressions would have been.
- Clipboard facilities provided by OilEd allowed (parts of) frames to be copied and pasted, making it easy to experiment with new definitions and to maintain a consistent modelling style. E.g., coenzymeA-requiring-oxidoreductase was built by copying nad-requiring-oxidoreductase and changing the constraint on the binds slot from nad to coenzymeA. The reasoner then automatically migrated the class from being a subclass of holoenzyme to being a subclass of coenzyme-requiring-enzyme.
- Class definitions can be as simple as possible yet as complex as necessary. Parts of the TaO are simply primitive frames and slots; other parts are very elaborate and exploit the full expressive power of the OIL language.

- In TAMBIS, the ontology is managed by an ontology server that makes full use of the class definitions, e.g., to classify user generated query classes. However, being able to deliver a static "snapshot" of the ontology in the form of an RDFS taxonomy has proved extremely convenient when working with collaborators who are building ontologies that are in fact simple taxonomies, such as the *Gene Ontology* [Ashburner *et al.*, 2000].

5 Conclusion

Ontologies are useful in a range of applications, where they provide a source of precisely defined terms that can be communicated across people and applications. We have used as an example, the initial development of a molecular biology and bioinformatics ontology. Examples from this case study have been used to demonstrate the utility of OIL's integration of features from frame and DL languages. It can be seen from the case study that OIL can support a cyclical ontology development, where incremental moves are made from a primitive, asserted taxonomy to one where concepts are rich with properties. These properties can be used to add richness to the ontology (from inferred knowledge), as well as ensuring the logical consistency and satisfiability of the ontology. Thus, the use of reasoning can be seen to be important for the design and management of ontologies during their development.

OilEd is a prototype development environment for OIL, designed to test and demonstrate novel ideas, and it still lacks many features that would be required of a fully-fledged ontology development environment, e.g., it provides no support for versioning, or for working with multiple ontologies. It is likely that during the development of the TaO that other, or fragments of other, ontologies will be imported into the TaO. Moreover, the reasoning support provided by the FaCT system is incomplete for OIL extended with concrete datatypes and individuals, and does not include additional services such as explanation. Thus, the definitions used for atom and the charge on ions is not used in constructing the classification. Explanation has potential use in both the development and use of a bio-ontology. During development, it will obviously be useful to have explanations of why a concept was unsatisfiable according to the current model. It is a goal for a bio-ontology, such as TaO, to be used in the analysis of novel biological macromolecules. Certain bioinformatics analyses can describe the properties of such molecules. If these could be cast in terms of the TaO, novel concepts generated by such analyses could be classified in the TaO. the use of explanation could significantly guide the use of such analyses.

During this case study, we have presented OIL and OilEd, an ontology editor that has an easy to use frame interface, yet at the same time allows users to exploit the full power of an expressive ontology language (OIL/DAML+OIL). We have also shown how OilEd uses reasoning to support ontology design and maintenance, and presented a case study illustrating how this facility can be used to develop ontologies that describe their domains in more detail and with greater fidelity.

Acknowledgements: Robert Stevens is supported by BBSRC/EPSRC grant 4/B1012090 and Sean Bechhofer is supported by EPSRC grant GR/M75426.

References

[Altman *et al.*, 1999] R. Altman, M. Bada, X.J. Chai, M. Whirl Carillo, R.O. Chen, and N.F. Abernethy. RiboWeb: An Ontology-Based System for Collaborative Molecular Biology. *IEEE Intelligent Systems*, 14(5):68–76, 1999.

[Ashburner *et al.*, 2000] M. Ashburner *et al.* Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, 1991.

[Baker *et al.*, 1998] P.G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview. In *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology*, pages 25–34. AAAI Press, June 28-July 1, 1998 1998.

[Baker *et al.*, 1999] P. Baker *et al.* An ontology for bioinformatics applications. *Bioinformatics*, 15(6):510–520, 1999.

[Brickley and Guha, 2000] D. Brickley and V.R. Guha. Resource description framework schema specification 1.0. W3C Candidate Recommendation, 2000. `http://www.w3.org/TR/rdf-schema`.

[Chaudhri *et al.*, 1998] V. K. Chaudhri *et al.* OKBC: A programmatic foundation for knowledge base interoperability. In *Proc. of AAAI-98*, 1998.

[Decker *et al.*, 2000] S. Decker *et al.* Knowledge representation on the web. In *Proc. of DL 2000*, pages 89–98, 2000.

[Doyle and Patil, 1991] J. Doyle and R. Patil. Two theses of knowledge representation. *Artificial Intelligence*, 48:261–297, 1991.

[Fensel *et al.*, 2000] D. Fensel *et al.* OIL in a nutshell. In *Proc. of EKAW-2000*, LNAI, 2000.

[Goble *et al.*, 2001] C. Goble *et al.* Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(2), 2001.

[Grosso *et al.*, 1999] W. E. Grosso *et al.* Knowledge modeling at the millennium (the design and evolution of protégé-2000). In *Proc. of KAW99*, 1999.

[Gruber, 1993] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In *Proc. of Int. Workshop on Formal Ontology*, 1993.

[Hendler and McGuinness, 2001] J. Hendler and D. L. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, jan 2001.

[Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180, 1999.

[Horrocks, 2000] I. Horrocks. Benchmark analysis with fact. In *Proc. TABLEAUX 2000*, pages 62–66, 2000.

[Karp *et al.*, 2000] P.D. Karp, M. Riley, M. Saier, I.T. Paulsen, S.M. Paley, and A. Pellegrini-Toole. The EcoCyc

and MetaCyc Databases. *Nucleic Acids Research*, 28:56–59, 2000.

[McGuinness and Borgida, 1995] D. McGuinness and A. Borgida. Explaining subsumption in description logics. In *Proc. of IJCAI-95*, pages 816–821, 1995.

[McGuinness *et al.*, 2000] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proc. of KR-00*, 2000.

[McGuinness, 1998] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS-98*, 1998.

[Rector *et al.*, 1997] A. Rector *et al.* The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.

[Staab and Maedche, 2000] S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In *Proc. of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods*, 2000.

[Stevens *et al.*, 2001] R. Stevens, C. A. Goble, and S. Bechhofer. Ontology-based knowledge representation for bioinformatics. *Briefings in Bioinformatics*, 2001.

[The Gene Ontology Consortium, 2000] The Gene Ontology Consortium. Gene Ontology: Tool for the Unification of Biology. *Nature Genetics*, 25:25–29, 2000.

[Uschold and Grüninger, 1996] M. Uschold and M. Grüninger. Ontologies: Principles, methods and applications. *K. Eng. Review*, 11(2):93–136, 1996.

[van Heijst *et al.*, 1997] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in KBS development. *Int. J. of Human-Computer Studies*, 46(2/3):183–292, 1997.

Ontology Merging for Federated Ontologies on the Semantic Web

Gerd Stumme

Institute for Applied Computer Science and
Formal Description Methods (AIFB)
University of Karlsruhe
D-76128 Karlsruhe, Germany
www.aifb.uni-karlsruhe.de/WBS/gst

Alexander Maedche

FZI Research Center
for Information Technologies
Haid-und-Neu-Strasse 10-14
D–76131 Karlsruhe, Germany
www.fzi.de/wim

Abstract

One of the core challenges for the Semantic Web is the aspect of decentralization. Local structures can be modeled by ontologies. However, in order to support global communication and knowledge exchange, mechanisms have to be developed for integrating the local systems. We adopt the database approach of autonomous federated database systems and consider an architecture for federated ontologies for the Semantic Web as starting point of our work.

We identify the need for merging specific ontologies for developing federated, but still autonomous web systems. We present the method FCA–MERGE for merging ontologies following a bottom-up approach which offers a structural description of the merging process. The method is guided by application-specific instances of the given source ontologies that are to be merged. We apply techniques from natural language processing and formal concept analysis to derive a lattice of concepts as a structural result of FCA–MERGE. The generated result is then explored and transformed into the merged ontology with human interaction.

1 Introduction

The current WWW is a great success with respect to the amount of stored documents and the number of users. One of the main reasons for the success of the current WWW is the principle of *decentralization* [Berners-Lee, 1999]. Currently the Semantic Web, developed as a "metaweb" for the WWW, is being established by standards for syntax (e. g. XML) and semantics (RDF(S), DAML+OIL, etc.). Ontologies have been established for knowledge sharing and are widely used as a means for conceptually structuring domains of interest. One of the core challenges for the Semantic Web is the aspect of decentralization.[1] Local structures can be modeled by ontologies. However, in order to support global communication and knowledge exchange, mechanisms have to be developed for integrating the local systems.

[1] cf. http://www.w3.org/DesignIssues/Principles.html

A number of proposals are available from the database community for developing multi-database systems and, more specific, federated database systems, that resemble the decentralized structures required in the Semantic Web. We adopt the database approach of federated databases and consider an architecture for federated ontologies on the Semantic Web as motivation and starting point of our work.

A bottleneck for federated ontologies in the Semantic Web is the process of integrating or merging specific ontologies. The process of *ontology merging* takes as input two (or more) source ontologies and returns a merged ontology based on the given source ontologies. Manual ontology merging using conventional editing tools without support is difficult, labor intensive and error prone. Therefore, several systems and frameworks for supporting the knowledge engineer in the ontology merging task have recently been proposed [Hovy, 1998; Chalupsky, 2000; Noy and Musen, 2000; McGuinness *et al*, 2000]. The approaches rely on syntactic and semantic matching heuristics which are derived from the behavior of ontology engineers when confronted with the task of merging ontologies, i. e. human behaviour is simulated. Although some of them locally use different kinds of logics for comparisons, these approaches do not offer a structural description of the global merging process.

We propose the new method FCA–MERGE for merging ontologies following a bottom-up approach which offers a global structural description of the merging process. For the source ontologies, it extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances we apply mathematically founded techniques taken from *Formal Concept Analysis* [Wille, 1982; Ganter and Wille, 1999] to derive a lattice of concepts as a structural result of FCA–MERGE. The produced result is explored and transformed to the merged ontology by the ontology engineer. The extraction of instances from text documents circumvents the problem that in most applications there are no objects which are simultaneously instances of the source ontologies, and which could be used as a basis for identifying similar concepts.

The remainder of the paper is as follows. We start our paper introducing a generic architecture for federating ontologies for the Semantic Web in Section 2. There we also identify the need for merging specific ontologies for developing federated, autonomous systems.

We briefly introduce some basic definitions concentrating on a formal definition of what an ontology is and recall the basics of Formal Concept Analysis in Section 3. In Sections 4 to 6, we present our method FCA–MERGE for merging ontologies following a bottom-up approach which offers a global structural description of the merging process. We present our generic method for ontology merging in Section 4. Section 5 provides a detailed description of FCA–MERGE. Section 6 gives an overview over related work, and Section 7 summarizes the paper and concludes with an outlook on future work.

2 An Architecture for Federated Ontologies in the Semantic Web

Figure 1 depicts the 5–layer architecture of federated ontologies on the Semantic Web. It adopts the approach of [Sheth & Larsen, 1990] for federated databases.

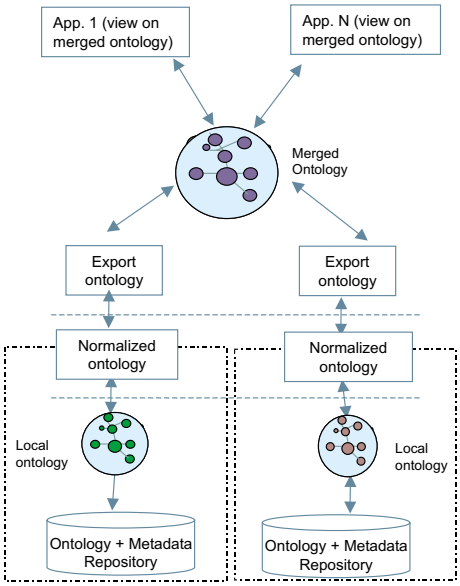


Figure 1: Architecture for Federated Ontologies

The architecture extends the standardized 3–layer schema architecture ANSI/SPARC with two additional layers. The adopted architecture mainly consists of:

1. local ontologies (the conceptual models of the autonomous systems), each of them with its specific underlying ontology/metadata repository or database,
2. normalized ontologies (transformation of the local ontologies into a common data model),
3. export ontologies (view on the normalized ontology that describes the relevant parts of the ontology for the federation),
4. one merged ontology (global ontology derived from the combination of the two export schemas), and
5. different applications in the upper layer (external schema layer), which use the merged ontology with their specific views on it.

In the following we will not go into further details of the organizational and architectural structure. As already mentioned, the following sections and the rest of this paper are dedicated to the task of generating a merged ontology from the two (or more) given export ontologies of the autonomous web systems.

3 Ontologies and Formal Concept Analysis

In this section, we briefly introduce some basic definitions. We thereby concentrate on a formal definition of what an ontology is and recall the basics of Formal Concept Analysis.

3.1 Ontologies

There is no common formal definition of what an ontology is. However, most approaches share a few core items: concepts, a hierarchical IS-A-relation, and further relations. For sake of generality, we do not discuss more specific features like constraints, functions, or axioms here. We formalize the core in the following way.

Definition: A *(core) ontology* is a tuple $\mathcal{O} := (\mathcal{C}, \texttt{is\_a}, \mathcal{R}, \sigma)$, where $\mathcal{C}$ is a set whose elements are called *concepts*, `is_a` is a partial order on $\mathcal{C}$ (i. e., a binary relation $\texttt{is\_a} \subseteq \mathcal{C} \times \mathcal{C}$ which is reflexive, transitive, and antisymmetric), $\mathcal{R}$ is a set whose elements are called *relation names* (or *relations* for short), and $\sigma\colon \mathcal{R} \to \mathcal{C}^+$ is a function which assigns to each relation name its arity.

As said above, the definition considers the core elements of most languages for ontology representation only. It is possible to map the definition to most types of ontology representation languages. Our implementation, for instance, is based on Frame Logic [Kifer *et al*, 1995]. Frame Logic has a well-founded semantics, but we do not refer to it in this paper.

3.2 Formal Concept Analysis

We recall the basics of Formal Concept Analysis (FCA) as far as they are needed for this paper. A more extensive overview is given in [Ganter and Wille, 1999]. To allow a mathematical description of concepts as being composed of extensions and intensions, FCA starts with a *formal context* defined as a triple $\mathbb{K} := (G, M, I)$, where G is a set of *objects*, M is a set of *attributes*, and I is a binary relation between G and M (i. e. $I \subseteq G \times M$). $(g, m) \in I$ is read "*object g has attribute m*".

Definition: For $A \subseteq G$, we define $A' := \{m \in M \mid \forall g \in A\colon (g, m) \in I\}$ and, for $B \subseteq M$, we define $B' := \{g \in G \mid \forall m \in B\colon (g, m) \in I\}$.

A *formal concept* of a formal context (G, M, I) is defined as a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The sets A and B are called the *extent* and the *intent* of the formal concept (A, B). The *subconcept–superconcept relation* is formalized by $(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2$ $(\Longleftrightarrow B_1 \supseteq B_2)$. The set of all formal concepts of a context $\mathbb{K}$ together with the partial order $\leq$ is always a complete lattice,[2] called the *concept lattice* of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$.

[2] I. e., for each set of formal concepts, there is always a greatest common subconcept and a least common superconcept.

A possible confusion might arise from the double use of the word 'concept' in FCA and in ontologies. This comes from the fact that FCA and ontologies are two models for the concept of 'concept' which arose independently. In order to distinguish both notions, *we will always refer to the FCA concepts as 'formal concepts'. The concepts in ontologies are referred to just as 'concepts' or as 'ontology concepts'.* There is no direct counter-part of formal concepts in ontologies. Ontology concepts are best compared to FCA attributes, as both can be considered as unary predicates on the set of objects.

4 Bottom-Up Ontology Merging

As said above, we propose a bottom-up approach for ontology merging. Our mechanism is based on application-specific instances of the two given ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ that are to be merged. The overall process of merging two[3] ontologies is depicted in Figure 2 and consists of three steps, namely *(i)* instance extraction and computing of two formal contexts $\mathbb{K}_1$ and $\mathbb{K}_2$, *(ii)* the FCA-MERGE core algorithm that derives a common context and computes a concept lattice, and *(iii)* the generation of the final merged ontology based on the concept lattice.

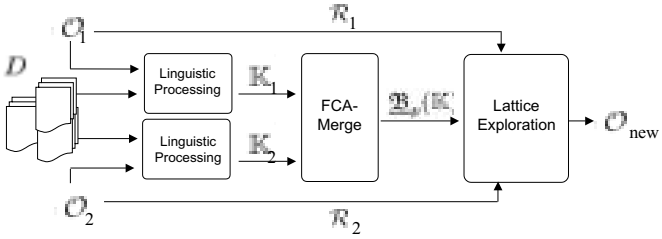


Figure 2: Ontology Merging Method

Our method takes as input data the two ontologies and a set D of natural language documents. The documents have to be relevant to both ontologies, so that the documents are described by the concepts contained in the ontology. The documents may be taken from the target application which requires the final merged ontology. From the documents in D, we *extract instances*. The mechanism for instance extraction is further described in Subsection 5.1. This automatic knowledge acquisition step returns, for each ontology, a formal context indicating which ontology concepts appear in which documents.

The extraction of the instances from documents is necessary because there are usually no instances which are already classified by both ontologies. However, if this situation is given, one can skip the first step and use the classification of the instances directly as input for the two formal contexts.

The second step of our ontology merging approach comprises the FCA–MERGE core algorithm. The core algorithm merges the two contexts and computes a concept lattice from the merged context using FCA techniques. More precisely, it computes a *pruned concept lattice* which has the same degree of detail as the two source ontologies. The techniques applied for generating the pruned concept lattice are described in Subsection 5.2 in more detail.

Instance extraction and the FCA–MERGE core algorithm are fully automatic. The final step of *deriving the merged ontology* from the concept lattice requires human interaction. Based on the pruned concept lattice and the sets of relation names $\mathcal{R}_1$ and $\mathcal{R}_2$, the ontology engineer creates the concepts and relations of the target ontology. We offer graphical means of the ontology engineering environment OntoEdit for supporting this process.

For obtaining good results, a few assumptions have to be met by the input data: Firstly, the documents have to be relevant to each of the source ontologies. A document from which no instance is extracted for each source ontology can be neglected for our task. Secondly, the documents have to cover all concepts from the source ontologies. Concepts which are not covered have to be treated manually after our merging procedure (or the set of documents has to be expanded). And last but not least, the documents must separate the concepts well enough. If two concepts which are considered as different always appear in the same documents, FCA-MERGE will map them to the same concept in the target ontology (unless this decision is overruled by the knowledge engineer). When this situation appears too often, the knowledge engineer might want to add more documents which further separate the concepts.

5 The FCA–MERGE Method

In this section, we discuss the three steps of FCA–MERGE in more detail. We illustrate FCA–MERGE with a small example taken from the tourism domain, where we have built several specific ontology-based information systems. Our general experiments are based on tourism ontologies that have been modeled in an ontology engineering seminar. Different ontologies have been modeled for a given text corpus on the web, which is provided by a WWW provider for tourist information.[4] The corpus describes actual objects, like locations, accommodations, furnishings of accommodations, administrative information, and cultural events. For the scenario described here, we have selected two ontologies: The first ontology contains 67 concepts and 31 relations, and the second ontology contains 51 concepts and 22 relations. The underlying text corpus consists of 233 natural language documents taken from the WWW provider described above. For demonstration purposes, we restrict ourselves first to two very small subsets $\mathcal{O}_1$ and $\mathcal{O}_2$ of the two ontologies described above; and to 14 out of the 233 documents. These examples will be translated in English. In Subsection 5.3, we provide some examples from the merging of the larger ontologies.

5.1 Linguistic Analysis and Context Generation

The aim of this first step is to generate, for each ontology $\mathcal{O}_i, i \in \{1, 2\}$, a formal context $\mathbb{K}_i := (G_i, M_i, I_i)$. The set of documents D is taken as object set ($G_i := D$), and the set of concepts is taken as attribute set ($M_i := \mathcal{C}_i$). While these

[3] The approach can easily be extended for merging n instead of two ontologies simultaneously.

[4] URL: http://www.all-in-all.com

I_1	Vacation	Hotel	Event	Concert	Root
doc1	×	×	×	×	×
doc2	×	×	×	×	×
doc3	×	×	×		×
doc4	×	×	×	×	×
doc5			×	×	×
doc6		×	×	×	×
doc7		×			×
doc8	×	×	×	×	×
doc9	×	×	×		×
doc10	×	×	×		×
doc11	×	×	×	×	×
doc12		×			×
doc13		×	×	×	×
doc14	×	×	×		×

I_2	Hotel	Accommodation	Musical	Root
doc1	×	×	×	×
doc2	×	×		×
doc3	×	×	×	×
doc4	×	×	×	×
doc5			×	×
doc6	×	×	×	×
doc7	×	×		×
doc8	×	×	×	×
doc9	×	×		×
doc10	×	×		×
doc11	×	×	×	×
doc12	×	×		×
doc13	×	×	×	×
doc14	×	×		×

Figure 3: The contexts $\mathbb{K}_1$ and $\mathbb{K}_2$ as result of the first step

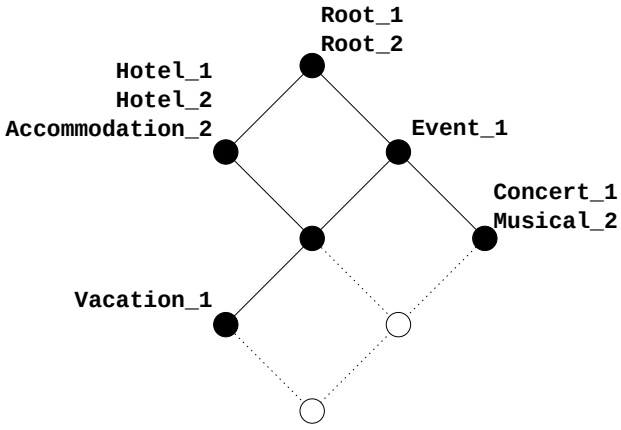


Figure 4: The pruned concept lattice

sets come for free, the difficult step is generating the binary relation I_i. The relation $(g, m) \in I_i$ shall hold whenever document g contains an instance of m.

The computation uses linguistic techniques as described in the sequel. We conceive an information extraction-based approach for ontology-based extraction, which has been implemented on top of SMES (Saarbrücken Message Extraction System), a shallow text processor for German (cf. [Neumann *et al*, 1997]). The architecture of SMES comprises a *tokenizer* based on regular expressions, a *lexical analysis* component including a *word and a domain lexicon*, and a *chunk parser*. The tokenizer scans the text in order to identify boundaries of words and complex expressions like "$20.00" or "Mecklenburg–Vorpommern",[5] and to expand abbreviations.

The lexicon contains more than 120,000 stem entries and more than 12,000 subcategorization frames describing information used for lexical analysis and chunk parsing. Furthermore, the domain-specific part of the lexicon contains lexical entries that express natural language representations of concepts and relations. Lexical entries may refer to several concepts or relations, and one concept or relation may be referred to by several lexical entries.

Lexical analysis uses the lexicon to perform *(1)* morphological analysis, i. e. the identification of the canonical common stem of a set of related word forms and the analysis of compounds, *(2)* recognition of named entities, *(3)* part-of-speech tagging, and *(4)* retrieval of domain-specific information. While steps (1), (2), and (3) can be viewed as standard for information extraction approaches, step (4) is of specific interest for our instance extraction mechanism. This step associates single words or complex expressions with a concept from the ontology if a corresponding entry in the domain-specific part of the lexicon exists. For instance, the expression "Hotel Schwarzer Adler" is associated with the concept `Hotel`. If the concept `Hotel` is in ontology $\mathcal{O}_1$ and document g contains the expression "Hotel Schwarzer Adler", then the relation (g,`Hotel`)$\in I_1$ holds.

Finally, the transitivity of the `is_a`-relation is compiled into the formal context, i. e. $(g, m) \in I$ and m `is_a` n implies $(g, n) \in I$. This means that if (g,`Hotel`)$\in I_1$ holds and `Hotel is_a Accommodation`, then the document also describes an instance of the concept `Accommodation`: (g,`Accommodation`)$\in I_1$.

Figure 3 depicts the contexts $\mathbb{K}_1$ and $\mathbb{K}_2$ that have been generated from the documents for the small example ontologies. E. g., document `doc5` contains instances of the concepts `Event`, `Concert`, and `Root` of ontology $\mathcal{O}_1$, and `Musical` and `Root` of ontology $\mathcal{O}_2$. All other documents contain some information on hotels, as they contain instances of the concept `Hotel` both in $\mathcal{O}_1$ and in $\mathcal{O}_2$.

5.2 Generating the Pruned Concept Lattice

The second step takes as input the two formal contexts $\mathbb{K}_1$ and $\mathbb{K}_2$ which were generated in the last step, and returns a *pruned concept lattice* (see below), which will be used as input in the next step.

First we merge the two formal contexts into a new formal context $\mathbb{K}$, from which we will derive the pruned concept lattice. Before merging the two formal contexts, we have to disambiguate the attribute sets, since $\mathcal{C}_1$ and $\mathcal{C}_2$ may contain the same concepts: Let $\widetilde{M}_i := \{(m, i) \mid m \in M_i\}$, for $i \in \{1, 2\}$. The indexation of the concepts allows the possibility that the same concept exists in both ontologies, but is treated differently. For instance, a `Campground` may be considered as an `Accommodation` in the first ontology, but not in the second one. Then the merged formal context is obtained by $\mathbb{K} := (G, M, I)$ with $G := D$, $M := \widetilde{M}_1 \cup \widetilde{M}_2$, and $(g, (m, i)) \in I :\Leftrightarrow (g, m) \in I_i$.

We will not compute the whole concept lattice of $\mathbb{K}$, as it would provide too many too specific concepts. We restrict the computation to those formal concepts which are above at least one formal concept generated by an (ontology) concept of the source ontologies. This assures that we remain within the range of specificity of the source ontologies. More precisely, the *pruned concept lattice* is given by $\underline{\mathfrak{B}}_p(\mathbb{K}) := \{(A, B) \in \underline{\mathfrak{B}}(\mathbb{K}) \mid \exists m \in M\colon (\{m\}', \{m\}'') \leq (A, B)\}$ (with $\cdot'$ as defined in Section 3.2).

For our example, the pruned concept lattice is shown in Figure 4. It consists of six formal concepts. Two formal con-

[5] a region in the north east of Germany

cepts of the total concept lattice are pruned since they are too specific compared to the two source ontologies. In the diagram, each formal concept is represented by a node. The empty nodes are the pruned concepts and are usually hidden from the user. A concept is a subconcept of another one if and only if it can be reached by a descending path. The intent of a formal concept consists of all attributes (i. e., in our application, the ontology concepts) which are attached to the formal concept or to one of its superconcepts. As we are not interested in the document names, the extents of the contexts are not visualized in this diagram.

The computation of the pruned concept lattice is done with the algorithm TITANIC [Stumme *et al*, 2000]. It is modified to allow the pruning. The modified algorithm is described below.

Compared to other algorithms for computing concept lattices, TITANIC has — for our purpose — the advantage that it computes the formal concepts via their *key sets* (or *minimal generators*). A key set is a minimal description of a formal concept:

Definition 1 *$K \subseteq M$ is a key set for the formal concept (A, B) if and only if $(K', K'') = (A, B)$ and $(X', X'') \neq (A, B)$ for all $X \subseteq K$ with $X \neq K$.*[6]

In our application, key sets serve two purposes. Firstly, they indicate if the generated formal concept gives rise to a new concept in the target ontology or not. A concept is new if and only if it has no key sets of cardinality one. Secondly, the key sets of cardinality two or more can be used as generic names for new concepts and they indicate the arity of new relations.

The TITANIC Algorithm. We recall the algorithm TITANIC and discuss how it is modified to compute the pruned concept lattice. In the following, we will use the composed function $\cdot'' : \mathfrak{P}(M) \to \mathfrak{P}(M)$ which is a closure operator on M (i. e., it is extensive, monotonous, and idempotent). The related closure system (i. e., the set of all $B \subseteq M$ with $B'' = B$) is exactly the set of the intents of all concepts of the context. The structure of the concept lattice is already determined by this closure system. Hence we restrict ourselves to the computation of all concept intents in the sequel. The computation makes extensive use of the following *support* function:

Definition 2 *The* support *of $X \subseteq M$ is defined by*

$$s(X) := \frac{|X'|}{|G|} \ .$$

We follow a pruning strategy given in [Agrawal and Srikant, 1994]. Originally this strategy was presented as a heuristic for determining all frequent sets only (i. e., all sets with supports above a user-defined threshold). The algorithm traverses the powerset of M in a level-wise manner. At the kth iteration, all subsets of M with cardinality k (called *k-sets*) are considered, unless we know in advance that they cannot be key sets.

[6] In other words: K generates the formal concept (A, B).

The pseudo-code of the modified TITANIC algorithm is given in Algorithm 1. A list of notations is provided in Table 1.

Algorithm 1 TITANIC

1) $\emptyset.s \leftarrow 1$;
2) $\mathcal{K}_0 \leftarrow \{\emptyset\}$;
3) $k \leftarrow 1$;
4) **forall** $m \in M$ **do** $\{m\}.p\_s \leftarrow 1$;
5) $\mathcal{C} \leftarrow \{\{m\} \mid m \in M\}$;
6) **loop begin**
7) COUNT$(\mathcal{C})$;
8) $\mathcal{K}_k \leftarrow \{X \in \mathcal{C} \mid X.s \neq X.p\_s$ and $(k = 1$ or $\exists m \in M : X \subseteq m.\text{closure})\}$;
9) **forall** $X \in \mathcal{K}_k$ **do** $X.\text{closure} \leftarrow$ CLOSURE(X);
10) **if** $\mathcal{K}_k = \emptyset$ **then exit loop** ;
11) $k{+}{+}$;
12) $\mathcal{C} \leftarrow$ TITANIC-GEN$(\mathcal{K}_{k-1})$;
13) **end loop** ;
14) **return** $\bigcup_{i=0}^{k-1} \{X.\text{closure} \mid X \in \mathcal{K}_i\}$.

Table 1: Notations used in TITANIC

k	is the counter which indicates the current iteration. In the kth iteration, all key k-sets are determined.
$\mathcal{K}_k$	contains after the kth iteration all key k-sets K together with their weight $K.s$ and their closure $K.\text{closure}$.
$\mathcal{C}$	stores the candidate k-sets C together with a counter $C.p\_s$ which stores the minimum of the weights of all $(k-1)$-subsets of C. The counter is used in step 8 to prune all non-key sets.

The algorithm starts with stating that the empty set is always a key set, and that its support is always equal to 1 (steps 1+2). Then all 1-sets are candidate sets by definition (steps 4+5). In later iterations, the candidate k-sets are determined by the function TITANIC-GEN (step 12/Algorithm 2) which is (except step 5) equivalent to the generating function of Apriori. (The result of step 5 will be used in step 8 of Algorithm 1 for pruning the non-key sets.)

Once the candidate k-sets are determined, the function COUNT$(\mathcal{X})$ is called to compute, for each $X \in \mathcal{X}$, the support of X. It is stored in the variable $X.s$ (step 7).

In step 8 of Algorithm 1, the second condition prunes all candidate k-sets which are out of the range of the two source ontologies. I. e., it implements the condition of the definition of the pruned concept lattice $\underline{\mathfrak{B}}_p(\mathbb{K})$. This additional condition makes the difference to the algorithm presented in [Stumme *et al*, 2000]. The first condition in step 8 prunes all candidate k-sets which are not key sets according to Proposition 1.

Proposition 1 ([Stumme *et al*, 2000]) *$X \subseteq M$ is a key set if and only if $s(X) \neq \min_{m \in X}(s(X \setminus \{m\}))$.*

For the remaining sets (which are now known to be key sets) their closures are computed (step 9). The CLOSURE

Algorithm 2 TITANIC-GEN

We assume that there is a total order $>$ on M.

Input: $\mathcal{K}_{k-1}$, the set of key $(k-1)$-sets K with their support $K.s$.

Output: $\mathcal{C}$, the set of candidate k-sets C
with the values $C.p\_s := \min\{s(C\setminus\{m\} \mid m \in C\}$.

The variables p_s assigned to the sets $\{p_1, \ldots, p_k\}$ which are generated in step 1 are initialized by $\{p_1, \ldots, p_k\}.p\_s \leftarrow 1$.

1) $\mathcal{C} \leftarrow \{\{p_1, \ldots, p_k\} \mid i < j \Rightarrow p_i < p_j,$
$\{p_1, \ldots, p_{k-2}, p_{k-1}\},$
$\{p_1, \ldots, p_{k-2}, p_k\} \in \mathcal{K}_{k-1}\}$;
2) **forall** $X \in \mathcal{C}$ **do begin**
3) **forall** $(k-1)$-subsets S of X **do begin**
4) **if** $S \notin \mathcal{K}_{k-1}$ **then begin** $\mathcal{C} \leftarrow \mathcal{C} \setminus \{X\}$;
exit forall ; **end**;
5) $X.p\_s \leftarrow \min(X.p\_s, S.s)$;
6) **end**;
7) **end**;
8) **return** $\mathcal{C}$.

Algorithm 3 CLOSURE(X) for $X \in \mathcal{K}_{k-1}$

1) $Y \leftarrow X$;
2) **forall** $m \in X$ **do** $Y \leftarrow Y \cup (X \setminus \{m\})$.closure;
3) **forall** $m \in M \setminus Y$ **do begin**
4) **if** $X \cup \{m\} \in \mathcal{C}$ **then** $s \leftarrow (X \cup \{m\}).s$
5) **else** $s \leftarrow \min\{K.s \mid K \in \mathcal{K},\ K \subseteq X \cup \{m\}\}$;
6) **if** $s = X.s$ **then** $Y \leftarrow Y \cup \{m\}$
7) **end**;
8) **return** Y.

function (Algorithm 3) is a straight-forward implementation of Proposition 2 (beside an additional optimization (step 2)).

Proposition 2 ([Stumme *et al*, 2000])

1. *Let $X \subseteq M$. Then*

$$h(X) = X \cup \{m \in M \setminus X \mid s(X) = s(X \cup \{m\})\} \ .$$

2. *If X is not a key set, then*

$$s(X) = \min\{s(K) \mid K \in \mathcal{K}, K \subseteq X\}$$

where $\mathcal{K}$ is the set of all key sets.

Algorithm 1 terminates, if there are no key k-sets left (step 10+14). Otherwise the next iteration begins (steps 11+12).

5.3 Generating the new Ontology from the Concept Lattice

While the previous steps (instance extraction, context derivation, context merging, and TITANIC) are fully automatic, the derivation of the merged ontology from the concept lattice requires human interaction, since it heavily relies on background knowledge of the domain expert.

The result from the last step is a pruned concept lattice. From it we have to derive the target ontology. Each of the formal concepts of the pruned concept lattice is a candidate for a concept, a relation, or a new subsumption in the target ontology. There is a number of queries which may be used to focus on the most relevant parts of the pruned concept lattice. We discuss these queries after the description of the general strategy — which follows now. Of course, most of the technical details are hidden from the user.

As the documents are not needed for the generation of the target ontology, we restrict our attention to the intents of the formal concepts, which are sets of (ontology) concepts of the source ontologies. For each formal concept of the pruned concept lattice, we analyze the related key sets. For each formal concept, the following cases can be distinguished:

1. It has exactly one key set of cardinality 1.
2. It has two or more key sets of cardinality 1.
3. It has no key sets of cardinality 0 or 1.
4. It has the empty set as key set.[7]

The generation of the target ontology starts with all concepts being in one of the two first situations. The first case is the easiest: The formal concept is generated by exactly one ontology concept from one of the source ontologies. It can be included in the target ontology without interaction of the knowledge engineer. In our example, these are the two formal concepts labeled by `Vacation_1` and by `Event_1`.

In the second case, two or more concepts of the source ontologies generate the same formal concept. This indicates that the concepts should be merged into one concept in the target ontology. The user is asked which of the names to retain. In the example, this is the case for two formal concepts: The key sets {`Concert_1`} and {`Musical_2`} generate the same formal concept, and are thus suggested to be merged; and the key sets {`Hotel_1`}, {`Hotel_2`}, and {`Accommodation_2`} also generate the same formal concept.[8] The latter case is interesting, since it includes two concepts of the same ontology. This means that the set of documents does not provide enough details to separate these two concepts. Either the knowledge engineer decides to merge the concepts (for instance because he observes that the distinction is of no importance in the target application), or he adds them as separate concepts to the target ontology. If there are too many suggestions to merge concepts which should be distinguished, this is an indication that the set of documents was not large enough. In such a case, the user might want to re-launch FCA–MERGE with a larger set of documents.

When all formal concepts in the first two cases are dealt with, then all concepts from the source ontologies are included in the target ontology. Now, all relations from the two source ontologies are copied into the target ontology. Possible conflicts and duplicates have to be resolved by the ontology engineer.

In the next step, we deal with all formal concepts covered by the third case. They are all generated by at least two concepts from the source ontologies, and are candidates for new

[7] This implies (by the definition of key sets) that the formal concept does not have another key set.

[8] {`Root_1`} and {`Root_2`} are no key sets, as each of them has a subset (namely the empty set) generating the same formal concept.

ontology concepts or relations in the target ontology. The decision whether to add a concept or a relation to the target ontology (or to discard the suggestion) is a modeling decision, and is left to the user. The key sets provide suggestions either for the name of the new concept, or for the concepts which should be linked with the new relation. Only those key sets with minimal cardinality are considered, as they provide the shortest names for new concepts and minimal arities for new relations, resp.

For instance, the formal concept in the middle of Figure 4 has {Hotel_2, Event_1}, {Hotel_1, Event_1}, and {Accommodation_2, Event_1} as key sets. The user can now decide if to create a new concept with the default name HotelEvent (which is unlikely in this situation), or to create a new relation with arity (Hotel, Event), e. g., the relation organizesEvent.

Key sets of cardinality 2 serve yet another purpose: $\{m_1, m_2\}$ being a key set implies that neither m_1 is_a m_2 nor m_2 is_a m_1 currently hold. Thus when the user does not use a key set of cardinality 2 for generating a new concept or relation, she should check if it is reasonable to add one of the two subsumptions to the target ontology. This case does not show up in our small example. An example from the large ontologies is given at the end of the section.

There is exactly one formal concept in the fourth case (as the empty set is always a key set). This formal concept gives rise to a new largest concept in the target ontology, the Root concept. It is up to the knowledge engineer to accept or to reject this concept. Many ontology tools require the existence of such a largest concept. In our example, this is the formal concept labeled by Root_1 and Root_2.

Finally, the is_a order on the concepts of the target ontology can be derived automatically from the pruned concept lattice: If the concepts c_1 and c_2 are derived from the formal concepts (A_1, B_1) and (A_2, B_2), resp., then c_1 is_a c_2 if and only if $B_1 \supseteq B_2$ (or if explicitly modeled by the user based on a key set of cardinality 2).

Querying the pruned concept lattice. In order to support the knowledge engineer in the different steps, there is a number of queries for focusing his attention to the significant parts of the pruned concept lattice.

Two queries support the handling of the second case (in which different ontology concepts generate the same formal concept). The first is a list of all pairs $(m_1, m_2) \in \mathcal{C}_1 \times \mathcal{C}_2$ with $\{m_1\}' = \{m_2\}'$. It indicates which concepts from the different source ontologies should be merged.

In our small example, this list contains for instance the pair (Concert_1, Musical_2). In the larger application (which is based on the German language), pairs like (Zoo_1, Tierpark_2) and (Zoo_1, Tiergarten_2) are listed. We decided to merge Zoo [engl.: zoo] and Tierpark [zoo], but not Zoo and Tiergarten [zoological garden].

The second query returns, for ontology $\mathcal{O}_i$ with $i \in \{1, 2\}$, the list of pairs $(m_i, n_i) \in \mathcal{C}_i \times \mathcal{C}_i$ with $\{m_i\}' = \{n_i\}'$. It helps checking which concepts out of a single ontology might be subject to merge. The user might either conclude that some of these concept pairs can be merged because their differentiation is not necessary in the target application; or he might decide that the set of documents must be extended because it does not differentiate the concepts enough.

In the small example, the list for $\mathcal{O}_1$ contains only the pair (Hotel_1, Accommodation_1). In the larger application, we had additionally pairs like (Räumliches, Gebiet) and (Auto, Fortbewegungsmittel). For the target application, we merged Räumliches [spatial thing] and Gebiet [region], but not Auto [car] and Fortbewegungsmittel [means of travel].

The number of suggestions provided for the third situation can be quite high. There are three queries which present only the most significant formal concepts out of the pruned concepts. These queries can also be combined.

Firstly, one can fix an upper bound for the cardinality of the key sets. The lower the bound is, the fewer new concepts are presented. A typical value is 2, which allows to retain all concepts from the two source ontologies (as they are generated by key sets of cardinality 1), and to discover new binary relations between concepts from the different source ontologies, but no relations of higher arity. If one is interested in having exactly the old concepts and relations in the target ontology, and no suggestions for new concepts and relations, then the upper bound for the key set size is set to 1.

Secondly, one can fix a minimum support. This prunes all formal concepts where the cardinality of the extent is too low (compared to the overall number of documents). In Algorithm 1, this is achieved by adding the condition "[...] and $X.s \geq$ minsupp" to step 8. The default is no pruning, i. e., with a minimum support of 0 %. It is also possible to fix different minimum supports for different cardinalities of the key sets. The typical case is to set the minimum support to 0 % for key sets of cardinality 1, and to a higher percentage for key sets of higher cardinality. This way we retain all concepts from the source ontologies, and generate new concepts and relations only if they have a certain (statistical) significance.

Thirdly, one can consider only those key sets of cardinality 2 in which the two concepts come from one ontology each. This way, only those formal concepts are presented which give rise to concepts or relations linking the two source ontologies. This restriction is useful whenever the quality of each source ontology *per se* is known to be high, i. e., when there is no need to extend each of the source ontologies alone.

In the small example, there are no key sets with cardinality 3 or higher. The three key sets with cardinality 2 (as given above) all have a support of $\frac{11}{14} \approx 78.6\,\%$. In the larger application, we fixed 2 as upper bound for the cardinality of the key sets. We obtained key sets like (Telefon_1 [telephone], Öffentliche_Einrichtung_2 [public institution]) (support = 24.5 %), (Unterkunft_1 [accommodation], Fortbewegungsmittel_2 [means of travel]) (1.7 %), (Schloß_1 [castle], Bauwerk_2 [building]) (2.1 %), and (Zimmer_1 [room], Bibliothek_2 [library]) (2.1 %). The first gave rise to a new concept Telefonzelle [public phone], the second to a new binary relation hatVerkehrsanbindung [hasPublicTransportConnection], the third to a new subsumption Schloß is_a Bauwerk, and the fourth was discarded as meaningless.

6 Related Work

A first approach for supporting the merging of ontologies is described in [Hovy, 1998]. There, several heuristics are described for identifying corresponding concepts in different ontologies, e. g. comparing the names and the natural language definitions of two concepts, and checking the closeness of two concepts in the concept hierarchy.

The OntoMorph system [Chalupsky, 2000] offers two kinds of mechanisms for translating and merging ontologies: syntactic rewriting supports the translation between two different knowledge representation languages, semantic rewriting offers means for inference-based transformations. It explicitly allows to violate the preservation of semantics in trade-off for a more flexible transformation mechanism.

In [McGuinness *et al*, 2000] the Chimaera system is described. It provides support for merging of ontological terms from different sources, for checking the coverage and correctness of ontologies and for maintaining ontologies over time. Chimaera offers a broad collection of functions, but the underlying assumptions about structural properties of the ontologies at hand are not made explicit.

Prompt [Noy and Musen, 2000] is an algorithm for ontology merging and alignment embedded in Protégé 2000. It starts with the identification of matching class names. Based on this initial step an iterative approach is carried out for performing automatic updates, finding resulting conflicts, and making suggestions to remove these conflicts.

The tools described above offer extensive merging functionalities, most of them based on syntactic and semantic matching heuristics, which are derived from the behaviour of ontology engineers when confronted with the task of merging ontologies. OntoMorph and Chimarea use a description logics based approach that influences the merging process locally, e. g. checking subsumption relationships between terms. None of these approaches offers a structural description of the global merging process. FCA–MERGE can be regarded as complementary to existing work, offering a structural description of the overall merging process with an underlying mathematical framework.

There is also much related work in the database community, especially in the area of federated database systems. The work closest to our approach is described in [Schmitt and Saake, 1998] and [Conrad, 1997]. They apply Formal Concept Analysis to a related problem, namely database schema integration. As in our approach, a knowledge engineer has to interpret the results in order to make modeling decisions. Our technique differs in two points: There is no need of knowledge acquisition from a domain expert in the preprocessing phase; and it additionally suggests new concepts and relations for the target ontology.

7 Conclusion and Future Work

We have motivated our work with the issue of decentralization, one of the main challenges for the Semantic Web. We have adopted the database point of view and consider an architecture for federating ontologies in the Semantic Web as motivation of our work. We discussed especially the process of integrating or merging specific ontologies which is a bottleneck for federated ontologies in the Semantic Web.

In this paper we have presented FCA–MERGE, a bottom-up technique for merging ontologies based on a set of documents. We have described the three steps of the technique: the linguistic analysis of the texts which returns two formal contexts; the merging of the two contexts and the computation of the pruned concept lattice; and the semi-automatic ontology creation phase which supports the user in modeling the target ontology. The paper described the underlying assumptions and discussed the methodology.

Future work includes the closer integration of the FCA–MERGE method in the ontology engineering environment ONTOEDIT. In particular, we will offer views on the pruned concept lattice based on the queries described in Subsection 5.3. It is also planned to further refine our information-extraction based mechanism for extracting instances. This refinement goes hand in hand with further improvements concerning the connection between ontologies and natural language (cf. [Maedche *et al*, 2001]).

The evaluation of ontology merging is an open issue [Noy and Musen, 2000]. We plan to use FCA–MERGE to generate independently a set of merged ontologies (based on two given source ontologies). Comparing these merged ontologies using the standard information retrieval measures as proposed in [Noy and Musen, 2000] will allow us to evaluate the performance of FCA–MERGE.

On the theoretical side, an interesting open question is the extension of the formalism to features of specific ontology languages, like for instance functions or axioms. The question is (i) how they can be exploited for the merging process, and (ii) how new functions and axioms describing the interplay between the source ontologies can be generated for the target ontology.

Future work also includes the implementation of the framework of federated ontologies as introduced in Section 2. We refer the interested reader to the recently started EU-IST funded project OntoLogging[9], where the development and management of federated web systems consisting of multiple ontologies and associated knowledge bases will be studied and implemented.

Acknowledgements

This research was partially supported by DFG and BMBF.

References

[Agrawal and Srikant, 1994] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. VLDB Conf.*, 1994, 478–499 (Expanded version in IBM Report RJ9839)

[Chalupsky, 2000] H. Chalupsky: OntoMorph: A translation system for symbolic knowledge. *Proc. KR '00*, Breckenridge, CO, USA,471–482.

[Conrad, 1997] S. Conrad: *Föderierte Datenbanksysteme: Konzepte der Datenintegration.* Informatik-Lehrbuch, Springer, Berlin–Heidelberg 1997

[Ganter and Wille, 1999] B. Ganter, R. Wille: *Formal Concept Analysis: mathematical foundations*. Springer.

[9] http://www.ontologging.com

[Berners-Lee, 1999] T. Berners-Lee: *Weaving the Web*. Harper.

[Hovy, 1998] E. Hovy: Combining and standardizing large-scale, practical ontologies for machine translation and other uses. *Proc. 1st Intl. Conf. on Language Resources and Evaluation*, Granada.

[Kifer *et al*, 1995] M. Kifer, G. Lausen, J. Wu: Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* **42**(4), 741–843.

[Maedche *et al*, 2001] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. *SEmantic PortAL - The SEAL approach*. to appear: In Creating the Semantic Web. D. Fensel, J. Hendler, H. Lieberman, W. Wahlster (eds.) MIT Press, MA, Cambridge, 2001.

[McGuinness *et al*, 2000] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder: An environment for merging and testing large Oontologies. *Proc. KR '00*, 483–493.

[Neumann *et al*, 1997] G. Neumann, R. Backofen, J. Baur, M. Becker, C. Braun: An information extraction core system for real world German text processing. *Proc. ANLP-97*, Washington.

[Noy and Musen, 2000] N. Fridman Noy, M. A. Musen: PROMPT: algorithm and tool for automated ontology merging and alignment. *Proc. AAAI '00*, 450–455

[Schmitt and Saake, 1998] I. Schmitt, G. Saake: Merging inheritance hierarchies for database integration. *Proc. CoopIS'98*, IEEE Computer Science Press, 322–331.

[Stumme *et al*, 2000] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal: Fast computation of concept lattices using data mining techniques. *Proc. KRDB ´00*, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/, 129–139.

[Sheth & Larsen, 1990] Sheth, A. & Larsen, J. (1990). Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys*, 22(3).

[Wille, 1982] R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets*. Reidel, Dordrecht, 445–470.

A knowledge model to support inconsistency management when reasoning with shared knowledge

Valentina A.M. Tamma & Trevor J.M. Bench-Capon
Department of Computer Science
University of Liverpool
Chadwick Building
Peach Street
L69 7ZF Liverpool UK

Abstract

This paper presents and motivates an extended ontology knowledge model which represents explicitly semantic information about concepts. This knowledge model results from enriching the usual conceptual model with semantic information which precisely characterises the concept's properties and expected ambiguities, including which properties are prototypical of a concept and which are exceptional, the behaviour of properties over time and the degree of applicability of properties to subconcepts. This enriched conceptual model permits a precise characterisation of what is represented by class membership mechanisms and helps a knowledge engineer to determine, in a straightforward manner, the meta-properties holding for a concept. Meta-properties are recognised to be the main tool for a formal ontological analysis that allows building ontologies with a clean and untangled taxonomic structure. This enriched semantics can prove useful to describe what is known by agents in a multi-agent systems, as it facilitates the use of reasoning mechanisms on the knowledge that instantiate the ontology. These mechanisms can be used to solve ambiguities that can arise when heterogeneous agents have to interoperate in order to perform a task.

1 Introduction

In the last decade ontologies have moved out of the research environment and have become widely used in many expert system applications not only to support the representation of knowledge but also complex inferences and retrieval. [McGuinness, 2000]. The extensive application of ontologies to broader areas has affected the notion of what ontologies are: they now range from light-weight ontologies, that is taxonomies of non-faceted concepts to more sophisticated ones where not only concepts but also their properties and relationships are represented.

The size of ontologies has also increased dramatically, and it is not so unusual to have ontologies with thousands of concepts. Such huge ontologies are sometimes the efforts of many domain experts and are designed and maintained in distributed environments. For this reasons research efforts are now devoted to merging and integrating diverse ontologies [Pinto *et al.*, 1999].

Lastly, the growing use of ontologies in expert systems requires that ontologies provide a ground for the application of reasoning techniques that result in sophisticated inferences such as those used to check and maintain consistency in knowledge bases.

The interest in designing ontologies that can be easily integrated and provide a base for applying reasoning mechanisms has stressed the importance of suitable conceptual models for ontologies. Indeed, it has been made a point that the sharing of ontologies depends heavily on a precise semantic representation of the concepts and their properties [Fridman Noy and Musen, 1999; McGuinness, 2000; Tamma and Bench-Capon, 2000].

This paper presents and motivates a knowledge model for ontologies which extends the usual set of facets in the OKBC frame-base model [Chaudhri *et al.*, 1998] to encompass more semantic information concerning the concept, which consists of a precise characterisation of the concept's properties and expected ambiguities, including which properties are prototypical of a concept and which are exceptional, the behaviour of the property over time and the degree of applicability of properties to subconcepts. This enriched knowledge model aims to provide enough semantic information to deal with problems of semantic inconsistency that arise when reasoning with integrated ontologies.

The paper is organised as follows: section 2 and subsections presents the motivations for adding semantics to the conceptual model, section 3 presents the knowledge model applying the conceptual model while in section 4 the model with respect to the motivations is discussed. Section 5 discusses the representation of roles by using the knowledge model and section 6 provides an example of concept description using the knowledge model, nally, in section 7 conclusions are drawn and future research directions are illustrated in section 8.

2 Encompassing semantics in the conceptual model

The motivation for enriching semantically the ontology conceptual model draws on three distinct arguments that are analysed in the reminder of this section.

2.1 Nature of ontologies

The first argument is based on the nature of ontologies. It has been argued that an ontology is *"an explicit specification of a conceptualisation"* [Gruber, 1993]. In other words an ontology *explicitly* defines the type of concepts used to describe the abstract model of a phenomenon and the constraints on their use. [Studer *et al.*, 1998]. An ontology is an *a priori* account of the objects that are in a domain and the relationships modelling the structure of the world seen from a particular perspective. In order to provide such an account one has to understand the concepts that are in the domain, and this involves a number of things. First it involves knowing what can sensibly be said of a thing falling under a concept. This can be represented by describing concepts in terms of their properties, and by giving a full characterisation of these properties. Thus, when describing the concept *Bird* it is important to distinguish that some birds fly and others do not. A full understanding of a concept involves more than this, however: it is important to recognise which properties are *prototypical* [Rosch, 1975] for the class membership and, more importantly, which are the permitted exceptions. There are, however differences in how confident we can be that an arbitrary member of a class conforms to the prototype: it is a very rare mammal that lays eggs, whereas many types of well known birds do not fly. Understanding a concept also involves understanding how and which properties change over time. This dynamic behaviour also forms part of the domain conceptualisation and can help to identify the *meta-properties* holding for the concept.

2.2 Integrating diverse ontologies

The second argument concerns the integration of ontologies. Integrating ontologies involves identifying overlapping concepts and creating a new concept, usually by generalising the overlapping ones, that has all the properties of the originals and so can be easily mapped into each of them. Newly created concepts inherit properties, usually in the form of attributes, from each of the overlapping ones. That is, let us suppose that the concept C is present in *n* ontologies $O_1, O_2, \cdots, O_n$, although described by different properties. That is each ontology $O_i, i = 1, \cdots, n$ defines a concept $C_i, i = 1, \cdots, n$ such that $C_1 \approx C_2 \approx \cdots \approx C_n$ (where $\approx$ denotes that the concepts are *overlapping*). Each concept $C_i, i = 1, \cdots, n$ is described in terms of a set of properties $P_i^C, i = 1, \cdots, n$. The result of the integration of the n ontologies is another ontology defining the concept $C_{integrated}$ which is defined in terms of $\bigcup_{i=1}^{n} P_i^C$, where all the P_i^C have to be distinguished.

One of the key points for integrating diverse ontologies is providing methodologies for building ontologies whose taxonomic structure is clean and untangled in order to facilitate the understanding, comparison and integration of concepts. Several efforts are focussing on providing engineering principles to build ontologies, for example [Gómez-Pérez, 1998; 1999]. Another approach [Guarino and Welty, 2000a; 2000b] concentrates on providing means to perform an ontological analysis which gives prospects for better taxonomies. This analysis is based on on a rigorous analysis of the *ontological meta-properties* of taxonomic nodes, which are based on the philosophical notions of *unity, identity, rigidity* and *dependence* [Guarino and Welty, 2000c].

When the knowledge encompassed in ontologies built for different purposes needs to be integrated inconsistencies can become evident. Many types of ontological inconsistencies have been defined in the literature, for instance in [Visser *et al.*, 1998] and the ontology environments currently available try to deal with these inconsistencies, such as SMART [Fridman Noy and Musen, 1999] and CHIMAERA [McGuinness *et al.*, 2000]. Here we broadly classify inconsistencies in ontologies into two types: structural and semantic. We define structural inconsistencies as those that arise because of differences in the properties that describe a concept. Structural inconsistencies can be detected and resolved automatically with limited intervention from the domain expert. For example, a concept C can be defined in two different ontologies O_1 and O_2 in terms of an attribute A that is specified as taking values in two different domains D_1 in O_1 and D_2 in O_2, where $D_1 \subseteq D_2$. Structural inconsistencies can be detected and resolved automatically with limited intervention from the domain expert.

Semantic inconsistencies are caused by the knowledge content of diverse ontologies which differs both in semantics and in level of granularity of the representation. They affect those attributes that are actually representing concept features and not relations with other concepts. Semantic inconsistencies require a deeper knowledge on the domain. Examples of semantic inconsistencies can be found in [McGuinness *et al.*, 2000; Tamma and Bench-Capon, 2000]. Adding semantics to the concept descriptions can be beneficial in solving this latter type of conflict, because a richer concept description provides more scope to resolve possible inconsistencies.

2.3 Reasoning with ontologies

The last argument to support the addition of semantics to ontology conceptual models turns on the need to reason with the knowledge expressed in the ontologies. Indeed, when different ontologies are integrated, new concepts are created from the definitions of the existing ones. In such a case conflicts can arise when conflicting information is inherited from two or more general concepts and one tries to reason with these concepts. Inheriting conflicting properties in ontologies is not as problematic as inheriting conflicting rules in knowledge bases, since an ontology is only *providing the means for describing explicitly the conceptualisation behind the knowledge represented in a knowledge base* [Bernaras *et al.*, 1996]. Thus, in a concept's description conflicting properties can coexist. However, when one needs to reason with the knowledge in the ontology, conflicting properties can hinder the reasoning process. Furthermore, if the ontologies one wants to reason with have been developed in different moments and for diverse purposes, it is likely that problem of *implicit in-*

consistencies might arise. This kind of problem is quite similar to the semantic inconsistencies that have been de ned in previous section. Such a problem has been rst identi ed in the inheritance literature [Morgenstern, 1998] where the author distinguishes between *explicit inconsistencies* from the *implict* ones. Explicit inconsistencies arise when two concepts C_i and C_j are described in terms of explicitly con icting properties that is in terms of the same attribute which is associated with con icting values V and $\neg V$. Implicit inconsistencies arise when the properties are described by different attributes but with opposite meanings. Morgenstern [Morgenstern, 1998] has modi ed the (notorious) Touretzky's Nixon diamond [Touretzky, 1986] to show an example of implicit inconsistencies. Let us consider:

- `Nixon`→ `Republican` ;
- `Nixon`→ `Quaker` ;
- `Quaker`→ `Pacifist` ;
- `Republican`→ `Hawk` ;

The two concepts `Quaker` and `Republican` are described by two attributes *Paci st* and *Hawk* that have different names but are semantically related (one is the opposite of the other), as they both describe someone's attitude towards going to war. In this case extra semantic information on the properties, such as the extent to which the property applies to the members of the class, can be used to derive which property is more likely to apply to the situation at hand. Of course, such sophisticated assumptions cannot be made automatically and need to be at least validated by knowledge engineers.

3 Extended knowledge model

In this section we extend a frame-based [Minsky, 1992] knowledge model. This is a result of the enriched conceptual model where properties are characterised with respect to their behaviour in the concept description. The knowledge model is based on *classes*, *slots*, and *facets*. *Classes* correspond to concepts and are collections of objects sharing the same properties, hierarchically organised into a multiple inheritance hierarchy, linked by *IS-A* links. Classes are described in terms of *slots*, or attributes, that can either be sets or single values. A slot is described by a name, a domain, a value type and by a set of additional constraints, here called *facets*. Facets can contain the documentation for a slot, constrain the value type or the cardinality of a slot, and provide further information concerning the slot and the way in which the slot is to be inherited by the subclasses. The set of facets has been extended from that provided by OKBC [Chaudhri *et al.*, 1998] in order to encompass descriptions of the attribute and its behaviour in the concept description and changes over time. The facets we use are listed below and discussed in the next section:

- **Value**: It associates a value $v \in$ **Domain** with an attribute in order to represent a property. However, when the concept that is de ned is very high in the hierarchy (so high that any conclusion as to the attribute's value is not possible), then either **Value** = `Domain`or **Value** = `Subdomain`$\subset$ `Domain`;
- **Type of value**: The possible llers for this facet are *Prototypical, Inherited, Distinguishing*. An attribute's value is a *Prototypical* one if the value is true for any prototypical instance of the concept, but exceptions are permitted with a degree of softness expressed by the facet **Ranking**. An attribute's value can be *Inherited* from some super concept or it can be a *Distinguishing* value, that is a value that differentiates among siblings. Note that distinguishing values become inherited values for subclasses of the class;
- **Exceptions**: It can be either a single value or a subset of the domain. It indicates those values that are permitted in the concept description because in the domain, but deemed exceptional from a common sense viewpoint. The exceptional values are not only those which differ from the prototypical ones but also any value which is possible but highly unlikely;
- **Ranking**: An integer describing the degree of con - dence of the fact that the attribute takes the values peci- ed in the facet **Value**. It describe the class membership condition. The possible values are 1: *All*, 2: *Almost all*, 3: *Most*, 4: *Possible*, 5: *A Few*, 6: *Almost none*, 7: *None*. For example, in the description of the concept *Bird* the slot *Ability to Fly* takes value `Yes`with *Ranking* 3, since not all birds y;
- **Change frequency**: Its possible values are: *Regular, Once only, Volatile, Never*. This facet describes how often an attribute's value changes. If the information is set equal to *Regular* it means that the process is continuous (see section below), for instance the age of a person can be modelled as changing regularly; if set equal to *Once only* it indicates that only one change is possible, for example a person's date of birth changes only once. If the slot is set equal to *Never* it means that the value associated with the attribute cannot change, and nally *Volatile* indicates that the attribute's value can change more than once, for example people can change job more than once;
- **Event**: Describes conditions under which the value changes. It is the set $\{((E_j, S_j, V_j), R_j) | j = 1, \cdots, m\}$ where E_j is an event, S_j is the state of the pair attribute-value associated with a property, V_j de nes the event validity and R_j denotes whether the change is reversible or not. The semantics of this facet is explained in the section below.

4 Relating the extended knowledge model to the motivations

The knowledge model presented in the previous section is motivated by the the problems described in section 2. It is based on an enriched semantics that aims to provide a better understanding of the concepts and their properties by characterising their behaviour.

Concept properties are to be considered on three levels: *instance level*, *class-membership level* and *meta level*. Properties at *instance level* are those exhibited by all the instances

of a concept. They might specialise properties at *class-membership level*, which instead describe properties holding for the class. Properties at *meta level* have been mainly described in philosophy, such as *identity, unity, rigidity* and *dependency*. The proposed model permits the characterisation of concepts on the three distinct property levels, thus also considering the meta level which is the basis for the ontological analysis illustrated in [Guarino and Welty, 2000b]. Such an enriched model helps to characterise the meta properties holding for the concepts, thus providing knowledge engineers with an aid to perform the ontological analysis which is usually demanding to perform.

Furthermore, the enriched knowledge model forces knowledge engineers to make ontological commitments explicit. Indeed, real situations are information-rich complete events whose context is so rich that, as it has been argued by Searle [Searle, 1983], it can never be fully speci ed. Many assumptions about meaning and context are usually made when dealing with real situations [Rosch, 1999]. These assumptions are rarely formalised when real situations are represented in natural language but they have to be formalised in an ontology since they are part ontological commitments that have to be made explicit. Enriching the semantics of the attribute descriptions with things such as the behaviour of attributes over time or how properties are shared by the subclasses makes some of the more important assumptions explicit.

The enriched semantics is essential to solve the inconsistencies that arise either while integrating diverse ontologies or while reasoning with the integrated knowledge. By adding information on the attributes we are able to better measure the similarity between concepts, to disambiguate between concepts that *seem* similar while they are not, and we have means to infer which property is likely to hold for a concept that inherits inconsistent properties. The remainder of this section describes the additional facets and relates them to the discussion in section 2.

4.1 Behaviour over time

In the knowledge model the facets *Change frequency* and *Event* describe the behaviour of properties over time, which models the changes in properties that are permitted in the concept's description without changing the essence of the concept. The behaviour over time is closely related to establishing the *identity* of concept descriptions [Guarino and Welty, 2000b]. Describing the behaviour over time involves also distinguishing properties whose change is *reversible* from those whose change is *irreversible*.

Property changes over time are caused either by the natural passing of time or are triggered by speci c event occurrences. We need, therefore, to use a suitable temporal framework that permits us to reason with time and events. The model chosen to accommodate the representation of the changes is the *Event Calculus* [Kowalski and Sergot, 1986]. Event calculus deals with local event and time periods and provides the ability to reason about change in properties caused by a speci c event and also the ability to reason with incomplete information.

Changes of properties can be modelled as *processes* [Sowa, 2000]. Processes can be described in terms of their starting and ending points and of the changes that happen in between. We can distinguish between *continuous* and *discrete changes*, the former describing incremental changes that take place continuously while the latter describe changes occurring in discrete steps called *events*. Analogously we can de ne *continuous properties* those changing regularly over time, such as the age of a person, versus *discrete properties* which are characterised by an event which causes the property to change. If the value associated with change frequency is *Regular* then the process is continuous, if it is *Volatile* the process is discrete and if it is *Once only* the process is considered discrete and the triggering event is set equal to *time-point=T*.

Any regular occurrence of time can be, however, expressed in form of an event, since most of the forms of reasoning for continuous properties require discrete approximations. Therefore in the knowledge model presented in the next section, continuous properties are modelled as discrete properties where the event triggering the change in property is the passing of time from the instant t to the instant t'. Each change of property is represented by a set of quadruples $\{((E_j, S_j, V_j), R_j) | j = 1, \cdots, m\}$ where E_j is an event, S_j is the state of the pair attribute-value associated with a property, V_j de nes the event validity while R_j indicates whether the change in properties triggered by the event E_j is reversible or not. The model used to accommodate this representation of the changes adds reversibility to *Event Calculus*, where each triple (E_j, S_j, V_j) is interpreted either as *the concept is in the state S_j before the event E_j happens* or *the concept is in the state S_j after the event E_j happens* depending on the value associated with V_j. The interpretation is obtained from the semantics of the event calculus, where the former expression is represented as *Hold(before(E_j, S_j))* while the latter as *Hold(after(E_j, S_j))*.

The idea of modelling the permitted changes for a property is strictly related to the philosophical notion of *identity*. In particular, the knowledge model addresses the problem of modelling identity when time is involved, namely *identity through changes*, which is based on the common sense notion that an individual may remain the same while showing different properties at different times [Guarino and Welty, 2000a]. The knowledge model we propose explicitly distinguishes the properties that can change from those which cannot, and describes the changes in properties that an individual can be subjected to, while still being recognised as an instance of a certain concept.

The notion of changes through time is also important to establish whether a property is *rigid*. A *rigid property* is de ned in [Guarino *et al.*, 1994] as:

> a property that is essential to *all* its instances, i.e. $\forall x \phi(x) \rightarrow \Box \phi(x)$.

The interpretation that is usually given to *rigidity* is that if x is an instance of a concept C than x has to be an instance of C in every possible world. Here we restrict ourselves to one of these systems of possible worlds, that is time. By characterising the rigidity of a property in this speci c world we aim to provide knowledge engineers the means to reach a better understanding on the *necessary* and *suf cient* conditions for the class membership.

4.2 Ranking

Rankings are de ned as [Goldszmidt and Pearl, 1996]:

> Each world is ranked by a non-negative integer representing the degree of surprise associated with nding such a world.

We have borrowed the term to denote the degree of surprise in nding a world where the property P holding for a concept C does not hold for one of its subconcepts C'. The additional semantics encompassed in this facet is important to reason with statements that have different degrees of truth. Indeed there is a difference in asserting facts such as "Mammals give birth to live young" and "Bird y", the former is generally more believable than the latter, for which many more counterexamples can be found. The ability to distinguish facts whose truth holds with different degrees of strength is related to nding facts that are true in every possible world and therefore constitute *necessary truth*. The concept of necessary truth brings us back to establishing whether a property is rigid or not, in fact it can be assumed that the value associated with the *Ranking* facet together with the temporal information on the changes permitted for the property lead us to determine whether the property described by the slot is a rigid one. Rigid properties have often been interpreted as *essential* properties (*i.e.*, a property holding for an individual in every possible circumstance in which the individual exists), however, we have to note that a property might be essential to a member of a class without being essential for membership in that class. For example, being odd is an essential property of the number 5, but it is not essential for membership in the class of prime numbers.

The ability to evaluate the degree of truth of a property in a concept description is also related to the problem of reasoning with ontologies obtained by integration. In such a case, as mentioned in section 2.3 inconsistencies can arise if a concepts inherits con icting properties. In order to be able to reason with these con icts some assumptions have to be made, concerning on how likely it is that a certain property holds; the facet *Ranking* models this information by modelling a qualitative evaluation of how subclasses inherit the property. This estimate represents the common sense knowledge expressed by linguistic quanti ers such as *All, Almost all, Few, etc.*.

In case of con icts the property's degree of truth can be used to rank the possible alternatives following an approach similar to the non-monotonic reasoning one developed by [Goldszmidt and Pearl, 1996]: in case of more con icting properties holding for a concept description, properties are ordered according to the degree of truth, that is according to the the ller associated with the *Ranking* facet weighted by the *Degree of strength*. Therefore, a property holding for all the subclasses is considered to have a higher rank than one holding for few of the concept subclasses, but this ordering is adjusted by the relevance, as perceived by the knowledge engineer, of the property in the concept's description (*Degree of strength*). For example, to reason about birds ability to y, the attribute *species* is more relevant than the attribute *feather colour*. When reasoning with diverse ontologies, the *Degree of strength* represents the weight associated with the inheritance rule corresponding to the attribute.

This ordering of the con icting properties needs to be validated by the knowledge engineer, however, it re ects the common sense assumption that, when no speci c information is known, people assume that the most likely property holds for a concept.

4.3 Prototypes and exceptions

In order to get a full understanding of a concept it is not sufcient to list the set of properties generally recognised as describing a typical instance of the concept but we need to consider the expected exceptions. Here we partially take the cognitive view of prototypes and graded structures, which is also re ected by the information modelled in the facet *Ranking*. In this view all cognitive categories show gradients of membership which describe how well a particular subclass ts people's idea or image of the category to which the subclass belong [Rosch, 1975]. Prototypes are the subconcepts which best represent a category, while exceptions are those which are considered exceptional although still belong to the category. In other words all the suf cient conditions for class membership hold for prototypes. For example, let us consider the biological category *mammal*: a *monotreme* (a mammal who does not give birth to live young) is an example of an exception with respect to this attribute. Prototypes depend on the context; there is no universal prototype but there are several prototypes depending on the context, therefore a prototype for the category *mammal* could be *cat* if the context taken is that of *pets* but it is *lion* if the assumed context is *circus animal*. Ontologies typically presuppose context and this feature is a major source of dif culty when merging them.

For the purpose of building ontologies, distinguishing the prototypical properties from those describing exceptions increases the expressive power of the description. Such distinctions do not aim at establishing default values but rather to guarantee the ability to reason with incomplete or con icting concept descriptions.

The ability to distinguish between prototypes and exceptions helps to determine which properties are necessary and sufcient conditions for concept membership. In fact a property which is prototypical and that is also inherited by all the subconcepts (that is it has the facet *Ranking* set to *All*) becomes a natural candidate for a necessary condition. Prototypes, therefore, describe the subconcepts that best t the cognitive category represented by the concept *in the speci c context given by the ontology*. On the other hand, by describing which properties are exceptional, we provide a better description of the class membership criteria in that it permits to determine what are the properties that, although rarely hold for that concept, are still possible properties describing the cognitive category. Here, the term *exceptional* is used to indicate something that differs from what is normally thought to be a feature of the cognitive category and not only what differs from the prototype.

Also the information on prototype and exceptions can prove useful in dealing with inconsistencies arising from ontology integration. When no speci c information is made available on a concept and it inherits con icting properties, then we can assume that the prototypical properties hold for it.

The inclusion of prototypes in the knowledge model provides the grounds for the semi-automatic maintenance and evolution of ontologies by applying techniques developed in other fields such as machine learning.

5 Prospects for supporting roles

The notion of *role* is central to any modelling activities as much as those of *objects* and *relations*. A thorough discussion of roles goes beyond the scope of this paper, and roles are not supported yet in the knowledge model introduced in section 3. However, the extended semantics provided by the knowledge model presented above gives good prospects for supporting roles. In this section we provide some preliminary consideration and relate the additional facets with the main features of the role notion.

Despite its importance that has been highlighted in the literature [Guarino, 1992; Sowa, 1984], only few modelling languages permit the distinction between a *concept* and the *roles* it can play in the knowledge model. This difficulty is partially due to the lack of a single definition for *role*.

A definition of role that makes use of the formal meta-properties and includes also the definition given by Sowa [Sowa, 1984] is provided by Guarino and Welty. In [Guarino and Welty, 2000a] they define a role as:

> properties expressing the *part played* by one entity in an event, often exemplifying a particular relationship between two or more entities. All roles are *anti-rigid* and *dependent*... A property ϕ is said to be anti-rigid if it is not essential to *all* its instances, i.e. $\forall x \phi(x) \rightarrow \neg\Box\phi(x)$... A property ϕ is *(externally) dependent* on a property ψ if, for all its instances x, necessarily some instance of ψ must exist, which is not a part nor a constituent of x, i.e. $\forall x \Box(\phi(x) \rightarrow \exists y \psi(y) \wedge \neg P(y,x) \wedge \neg C(y,x))$.

In other words a concept is a role if its individuals stand in relation to other individuals, and they can enter or leave the extent of the concept without losing their identity. From this definition it emerges that the ability of recognising whether rigidity holds for some property ϕ is essential in order to distinguish whether ϕ is a role.

In [Steimann, 2000] the author presents a list of the features that have been associated in the literature with roles. Some of these features are conflicting and, as pointed out, no integrating definition has been made available. However, from the different definitions available it can be derived that the notion of role is inherently temporal, indeed roles are acquired and relinquished in dependence either of time or of a specific event. For example the object *person* acquires the role *teenager* if the person is between 11 and 19 years old, whereas a person becomes *student* when they enroll for a degree course. Moreover, from the list of features in [Steimann, 2000] it emerges that many of the characteristics of roles are time or event related, such as: an object may acquire and abandon roles dynamically, may play different roles simultaneously, or may play the same role several time, simultaneously, and the sequence in which roles may be acquired and relinquished can be subjected to restrictions.

For the aforementioned reasons ways of representing roles must be supported by some kind of time and event explicit representation. We believe that the knowledge model we have presented, although it does not encompass roles yet, provides sufficient semantics to model the dynamic features of roles, thanks to the explicit representation of time intervals which is used to model the attributes behaviour over time. Furthermore, the ability of modelling events, used to describe the possible causes in the state of an attribute, can be used to model the events that constrain the acquisition or the relinquishment of a role.

6 A modelling example

We now provide an example to illustrate how the previously described knowledge model can be used for modelling a concept in the ontology. The example is taken from the medical domain and we have chosen to model the concept of *blood pressure*. Blood pressure is represented here as an ordered pair (s, d) where *s* is the value of the *systolic pressure* while *d* is the value of the *diastolic pressure*. In modelling the concept of blood pressure we take into account that both the systolic and diastolic pressure can range between a minimum and a maximum value but that some values are more likely to be registered than others. Within the likely values we then distinguish the *prototypical* values, which are those registered for a healthy individual whose age is over 18, and the *exceptional* ones, which are those registered for people with pathologies such as hypertension or hypotension. The prototypical values are those considered normal, but they can change and we describe also the permitted changes and what events can trigger such changes. Prototypical pressure values usually change with age, but they can be altered depending on some specific events such as shock and haemorrhage (causing hypotension) or thrombosis and embolism (causing hypertension). Also conditions such as pregnancy can alter the normal readings.

Classes are denoted by the label **c**, slots by the label **s** and facets by the label **f**. Irreversible changes are denoted by I while reversible property changes are denoted by R.

c: `Circulatorysystem;`
s: `Bloodpressure`
f: `Domain`: [(0,0)-(300,200)];
f: `Value`: [(90,60)-(130,85)];
f: `Typeofvalue`: prototypical;
f: `Exceptions`: [(0,0)-(89,59)] ∪ [(131,86)-(300,200)];
f: `Ranking`: 3;
f: `Changefrequency`: Volatile;
f: `Event`: (Age=60,[(0,0)-(89,59)] ∪
∪ [(131,86)-(300,200)],after, I);
f: `Event`: (haemorrhage,[(0,0)-(89,59)],after, R);
f: `Event`: (shock,[(0,0)-(89,59)],after, R);
f: `Event`: (thrombosis,[(131,86)-(300,200)],after,R);
f: `Event`: (embolism,[(131,86)-(300,200)],after,R);
f: `Event`: (pregnancy,[(0,0)-(89,59)] ∪
∪ [(131,86)-(300,200)],after,R);

7 Conclusions

This paper has presented a knowledge model that extends the usual ontology frame-based model such as OKBC by explicitly representing additional information on the slot properties. This knowledge model results from a conceptual model which encompasses semantic information aiming to characterise the behaviour of properties in the concept description. We have motivated this enriched conceptual model by identifying three main categories of problems that require additional semantics in order to be solved.

The novelty of this extended knowledge model is that it explicitly represents the behaviour of attributes over time by describing the permitted changes in a property that are permitted for members of the concept. It also explicitly represents the class membership mechanism by associating with each slot a qualitative quanti er representing how properties are inherited by subconcepts. Finally, the model does not only describe the prototypical properties holding for a concept but also the exceptional ones.

We have also related the extended knowledge model to the formal ontological analysis by Guarino and Welty [Guarino and Welty, 2000b] which permits to build ontologies that have a cleaner taxonomic structure and so gives better prospects for maintenance and integration. Such a formal ontological analysis is usually dif cult to perform and we believe our knowledge model can help knowledge engineers to determine the meta-properties holding for the concept by forcing them to make the ontological commitments explicit.

A possible drawback of this approach is the high number of facets that need to lled when building ontology. We realise that this can make building an ontology from scratch even more time consuming but we believe that the outcomes in terms of better understanding of the concept and the role it plays in a context together with the guidance in determining the meta-properties at least balances the increased complexity of the task.

8 Future work

The extension of the knowledge model with with additional semantics opens several new research directions. Firstly, the role representation needs to be formalised in the knowledge model in order to represent also the roles hierarchical organisation [Steimann, 2000].

We also plan to use the semantics encompassed in the knowledge model to assist knowledge engineers in the tasks of merging and reasoning with diverse ontologies. To reach this goal we intent to introduce some form of temporal reasoning based on the event logics that is used extend the facets.

The description of attributes in terms of prototypical values gives us the possibility of exploring the application of machine learning techniques to ontologies, in order to extend them dynamically by learning new concept descriptions and placing them in the most appropriate position in the class hierarchy.

Acknowledgement

The PhD research presented in this paper was funded by BT plc. The authors are grateful to Ray Paton for providing the example. The authors would also wish to thank Floriana Grasso, Floriana Esposito and Donato Malerba for their invaluable comments on the paper.

References

[Bernaras *et al.*, 1996] A. Bernaras, I. Laresgoiti, and J. Corera. Building and reusing ontologies for electrical network applications. In *Proceedings of the 12th European Conference on Arti cial Intelligence (ECAI)* , pages 298–302, 1996.

[Chaudhri *et al.*, 1998] V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, and J.P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the Fifteenth American Conference on Arti cial Intelligence (AAAI-98)* , pages 600–607, Madison, Wisconsin, 1998. AAAI Press/The MIT Press.

[Fridman Noy and Musen, 1999] N. Fridman Noy and M.A. Musen. SMART: Automated support for ontology merging and alignment. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW)*, Banff, Canada, 1999.

[Goldszmidt and Pearl, 1996] M. Goldszmidt and J. Pearl. Qualitative probabilistic for default reasoning, belief revision, and causal modelling. *Arti cial Intelligence* , 84(1-2):57–112, 1996.

[Gómez-Pérez, 1998] A. Gómez-Pérez. Knowledge sharing and reuse. In J. Liebowitz, editor, *The Handbook of Applied Expert Systems*. CRC Pres LLC, 1998.

[Gómez-Pérez, 1999] A. Gómez-Pérez. Ontological engineering: A state of the art. *Expert Update*, 2(3):33–43, Autumn 1999.

[Gruber, 1993] T. R. Gruber. A translation approach to portable ontology speci cations. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Guarino and Welty, 2000a] N. Guarino and C. Welty. A formal ontology of properties. In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, volume LNAI 1937. Springer Verlag, 2000.

[Guarino and Welty, 2000b] N. Guarino and C. Welty. Identity, unity and individuality: Towards a formal toolkit for ontological analysis. In W. Horn, editor, *Proceedings of the 14th European Conference on Arti cial Intelligence (ECAI)*, Amsterdam, 2000. IOS Press.

[Guarino and Welty, 2000c] N. Guarino and C. Welty. Towards a methodology for ontology based model engineering. In *Proceedings of the ECOOP 2000 workshop on model engineering*, 2000.

[Guarino *et al.*, 1994] N. Guarino, M. Carrara, and P. Giaretta. An ontology of meta-level-categories. In *Principles of Knowledge representation and reasoning: Proceedings of the fourth international conference (KR94)*. Morgan Kaufmann, 1994.

[Guarino, 1992] N. Guarino. Concepts, attributes and aritrary relations. *Data and Knowledge Engineering*, 8:249–261, 1992.

[Kowalski and Sergot, 1986] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[McGuinness *et al.*, 2000] D.L. McGuinness, R.E. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceedings of KR-2000. Principles of Knowledge Representation and Reasoning.* Morgan-Kaufman, 2000.

[McGuinness, 2000] D.L. McGuinness. Conceptual modelling for distributed ontology environments. In *Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS 2000)*, 2000.

[Minsky, 1992] M. Minsky. A framework for representing knowledge. In A. Collins and E.H. Smith, editors, *Cognitive Science*, pages 191–215. Morgan Kaufmann, Los Altos, CA, 1992.

[Morgenstern, 1998] L. Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Arti cial Intelligence* , 103:1–34, 1998.

[Pinto *et al.*, 1999] H.S. Pinto, A. Gómez-Pérez, and J.P. Martins. Some issues on ontology integration. In V.R. Benjamins, editor, *Proceedings of the IJCAI'99 Workshop on Ontology and Problem-Solving Methods: Lesson learned and Future Trends*, volume 18, pages 7.1–7.11, Amsterdam, 1999. CEUR Publications.

[Rosch, 1975] E.H. Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104:192–233, 1975.

[Rosch, 1999] E.H. Rosch. Reclaiming concepts. *Journal of Consciousness Studies*, 6(11-12):61–77, 1999.

[Searle, 1983] J.R. Searle. *Intentionality*. Cambridge University Press, Cambridge, 1983.

[Sowa, 1984] J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Sowa, 2000] J.F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Paci c Grove, CA, 2000.

[Steimann, 2000] F. Steimann. On the representation of roles in object-oriented and conceptual modelling. *Data and Knowledge Engineering*, 35:83–106, 2000.

[Studer *et al.*, 1998] R. Studer, V.R. Benjamins, and D. Fensel. Knowledge engineering, principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[Tamma and Bench-Capon, 2000] V.A.M. Tamma and T.J.M Bench-Capon. Supporting inheritance mechanisms in ontology representation. In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, volume LNAI 1937, pages 140–155. Springer Verlag, 2000.

[Touretzky, 1986] D.S. Touretzky. *The Mayhematics of Inheritance Systems*. Morgan Kaufmann, 1986.

[Visser *et al.*, 1998] P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon, and M.J.R. Shave. Assessing heterogeneity by classifying ontology mismatches. In N. Guarino, editor, *Formal Ontology in Information Systems. Proceedings FOIS'98, Trento, Italy* , pages 148–182. IOS Press, 1998.

Ontology-Based Integration of Information — A Survey of Existing Approaches

H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner
Intelligent Systems Group, Center for Computing Technologies,
University of Bremen, P.O.B. 33 04 40, D-28334 Bremen, Germany
e-mail: {wache|vogele|visser|heiner|schuster|neumann|huebner}@tzi.de

Abstract

We review the use on ontologies for the integration of heterogeneous information sources. Based on an in-depth evaluation of existing approaches to this problem we discuss how ontologies are used to support the integration task. We evaluate and compare the languages used to represent the ontologies and the use of mappings between ontologies as well as to connect ontologies with information sources. We also enquire into ontology engineering methods and tools used to develop ontologies for information integration. Based on the results of our analysis we summarize the state-of-the-art in ontology-based information integration and name areas of further research activities.

1 Motivation

The so-called information society demands complete access to available information, which is often heterogeneous and distributed. In order to establish efficient information sharing, many technical problems have to be solved. First, a suitable information source must be located that might contain data needed for a given task. Finding suitable information sources is a problem addressed in the areas of information retrieval and information filtering [Belkin and Croft, 1992]. Once the information source has been found, access to the data therein has to be provided. This means that each of the information sources found in the first step have to work together with the system that is querying the information. The problem of bringing together heterogeous and distributed computer systems is known as *interoperability problem.*

Interoperability has to be provided on a technical and on an informational level. In short, information sharing not only needs to provide full accessibility to the data, it also requires that the accessed data may be processed and interpreted by the remote system. Problems that might arise owing to heterogeneity of the data are already well-known within the distributed database systems community (e.g. [Kim and Seo, 1991], [Kashyap and Sheth, 1996a]): *structural heterogeneity* (schematic heterogeneity) and *semantic heterogeneity* (data heterogeneity) [Kim and Seo, 1991]. Structural heterogeneity means that different information systems store their data in different structures. Semantic heterogeneity considers the contents of an information item and its intended meaning.

In order to achieve semantic interoperability in a heterogeneous information system, the *meaning* of the information that is interchanged has to be understood across the systems. Semantic conflicts occur whenever two contexts do not use the same interpretation of the information. Goh identifies three main causes for semantic heterogeneity [Goh, 1997]:

- *Confounding conflicts* occur when information items seem to have the same meaning, but differ in reality, e.g. owing to different temporal contexts.
- *Scaling conflicts* occur when different reference systems are used to measure a value. Examples are different currencies.
- *Naming conflicts* occur when naming schemes of information differ significantly. A frequent phenomenon is the presence of homonyms and synonyms.

The use of ontologies for the explication of implicit and hidden knowledge is a possible approach to overcome the problem of semantic heterogeneity. Uschold and Grüninger mention interoperability as a key application of ontologies, and many ontology-based approaches [Uschold and Grüniger, 1996] to information integration in order to achieve interoperability have been developed.

In this paper we present a survey of existing solutions with special focus on the use of ontologies in these approaches. We analyzed about 25 approaches to intelligent information integration including SIMS, TSIMMIS, OBSERVER, CARNOT, Infosleuth, KRAFT, PICSEL, DWQ, Ontobroker, SHOE and others with respect to the role and use of ontologies. Most of these systems use some notion of ontologies. We only consider these approaches. A further criterion is the focus of the approach on the integration of information sources. We therefore do not consider approaches to the integration of knowledge bases. We evaluate the remaining approaches according to four main criteria:

Use of Ontologies: The role and the architecture of the ontologies influence heavily the representation formalism of an ontology.

Ontology Representation: Depending on the use of the ontology, the representation capabilities differ from approach to approach.

Use of Mappings: In order to support the integration process the ontologies have to be linked to actual information. If several ontologies are used in an integration system, mapping between the ontologies are also important.

Ontology Engineering: Before an integration of information sources can begin the appropriate ontologies have to be acquired or to be selected for reuse. How does the integration approach support the acquisition or reuse of ontologies?

In the following we discuss these points on the basis of our experiences from the comparison of different systems. Doing this we will not consider single approaches, but rather refer to typical representatives. In section 2 we discuss the use of ontologies in different approaches and common ontology architectures. The use of different representations, i.e. different ontology languages, is discussed in section 3. Mappings used to connect ontologies to information sources and inter-ontology mappings are the topic of section 4, while section 5 covers methodologies and tool-support for the ontology engineering process. We conclude with a summary of the state-of-the-art and the direction for further research in the area of ontology-based information integration.

2 The Role of Ontologies

Initially, ontologies are introduced as an "explicit specification of a conceptualization" [Gruber, 1993]. Therefore, ontologies can be used in an integration task to describe the semantics of the information sources and to make the contents explicit (section 2.1). With respect to the integration of data sources, they can be used for the identification and association of semantically corresponding information concepts.

However, in several projects ontologies take over additional tasks. These tasks are discussed in section 2.2.

2.1 Content Explication

In nearly all ontology–based integration approaches ontologies are used for the explicit description of the information source semantics. But there are different way of how to employ the ontologies. In general, three different directions can be identified: *single ontology approaches*, *multiple ontologies approaches* and *hybrid approaches*. Figure 1 gives an overview of the three main architectures.

The integration based on a single ontology seems to be the simplest approach because it can be simulated by the other approaches. Some approaches provide a general framework where all three architectures can be implemented (e.g. DWQ [Calvanese *et al.*, 2001]). The following paragraphs give a brief overview of the three main ontology architectures.

Single Ontology approaches Single ontology approaches use one global ontology providing a shared vocabulary for the specification of the semantics (see fig. 1a). All information sources are related to one global ontology. A prominent approach of this kind of ontology integration is SIMS [Arens *et al.*, 1996]. SIMS model of the application domain includes a hierarchical terminological knowledge base with nodes representing objects, actions, and states. An independent model

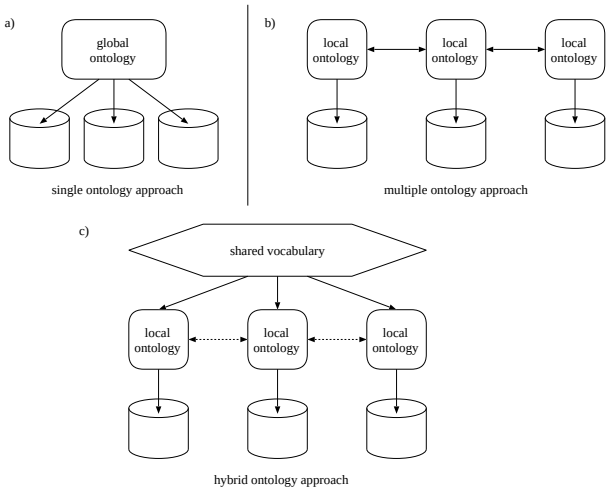


Figure 1: The three possible ways for using ontologies for content explication

of each information source must be described for this system by relating the objects of each source to the global domain model. The relationships clarify the semantics of the source objects and help to find semantically corresponding objects.

The global ontology can also be a combination of several specialized ontologies. A reason for the combination of several ontologies can be the modularization of a potentially large monolithic ontology. The combination is supported by ontology representation formalisms i.e. importing other ontology modules (cf. ONTOLINGUA [Gruber, 1993]).

Single ontology approaches can be applied to integration problems where all information sources to be integrated provide nearly the same view on a domain. But if one information source has a different view on a domain, e.g. by providing another level of granularity, finding the minimal ontology commitment [Gruber, 1995] becomes a difficult task. For example, if two information sources provide product specifications but refer to absolute heterogeneous product catalogues which categorize the products, the development of a global ontology which combines the different product catalogues becomes very difficult. Information sources with reference to similar product catalogues are much easier to integrate. Also, single ontology approaches are susceptible to changes in the information sources which can affect the conceptualization of the domain represented in the ontology. Depending on the nature of the changes in one information source it can imply changes in the global ontology and in the mappings to the other information sources. These disadvantages led to the development of multiple ontology approaches.

Multiple Ontologies In multiple ontology approaches, each information source is described by its own ontology (fig. 1b). For example, in OBSERVER [Mena *et al.*, 1996] the semantics of an information source is described by a separate ontology. In principle, the "source ontology" can be a combination of several other ontologies but it can not be assumed

that the different "source ontologies" share the same vocabulary.

At a first glance, the advantage of multiple ontology approaches seems to be that no common and minimal ontology commitment [Gruber, 1995] about one global ontology is needed. Each source ontology could be developed without respect to other sources or their ontologies — no common ontology with the agreement of all sources are needed. This ontology architecture can simplify the change, i.e. modifications in one information source or the adding and removing of sources. But in reality the lack of a common vocabulary makes it extremely difficult to compare different source ontologies. To overcome this problem, an additional representation formalism defining the inter-ontology mapping is provided (see 4.2). The inter-ontology mapping identifies semantically corresponding terms of different source ontologies, e.g. which terms are semantically equal or similar. But the mapping also has to consider different views on a domain e.g. different aggregation and granularity of the ontology concepts. We believe that in practice the inter-ontology mapping is very difficult to define, because of the many semantic heterogeneity problems which may occur.

Hybrid Approaches To overcome the drawbacks of the single or multiple ontology approaches, hybrid approaches were developed (Fig. 1c). Similar to multiple ontology approaches the semantics of each source is described by its own ontology. But in order to make the source ontologies comparable to each other they are built upon one global shared vocabulary [Goh, 1997; Wache *et al.*, 1999]. The shared vocabulary contains basic terms (the primitives) of a domain. In order to build complex terms of a source ontologies the primitives are combined by some operators. Because each term of a source ontology is based on the primitives, the terms become easier comparable than in multiple ontology approaches. Sometimes the shared vocabulary is also an ontology [Stuckenschmidt *et al.*, 2000b].

In hybrid approaches the interesting point is how the local ontologies are described, i.e. how the terms of the source ontology are described by the primitives of the shared vocabulary. In COIN [Goh, 1997] the local description of an information, the so-called context, is simply an attribute value vector. The terms for the context stems from the common shared vocabulary and the data itself. In MECOTA [Wache *et al.*, 1999], each source information is annotated by a label which indicates the semantics of the information. The label combines the primitive terms from the shared vocabulary. The combination operators are similar to the operators known from the description logics, but are extended for the special requirements resulting from integration of sources, e.g. by an operator which indicates that an information aggregates several different information items (e.g. a street name together with number). In BUSTER [Stuckenschmidt *et al.*, 2000b], the shared vocabulary is a (general) ontology, which covers all possible refinements. E.g. the general ontology defines the attribute value ranges of its concepts. A source ontology is one (partial) refinement of the general ontology, e.g. restricts the value range of some attributes. Since the source ontologies only use the vocabulary of the general ontology, they remain comparable.

The advantage of a hybrid approach is that new sources can easily be added without the need of modification in the mappings or in the shared vocabulary. It also supports the acquisition and evolution of ontologies. The use of a shared vocabulary makes the source ontologies comparable and avoids the disadvantages of multiple ontology approaches. The drawback of hybrid approaches however, existing ontologies cannot be reused easily, but have to be re-developed from scratch, because all source ontologies have to refer to the shared vocabulary.

The following table summarizes the benefits and drawbacks of the different ontology approaches:

	Single Ontology Approaches	Multiple Ontology Approaches	Hybrid Ontology Approaches
implementation effort	straightforward	costly	reasonable
semantic heterogeneity	similar view of a domain	supports heterogeneous views	supports heterogeneous views
adding/ removing of sources	need for some adaption in the global ontology	providing a new source ontology; relating to other ontologies	providing a new source ontology;
comparing of multiple ontologies	—	difficult because of the lack of a common vocabulary	simple because ontologies use a common vocabulary

Table 1: Benefits and drawbacks of the different ontology-based integration approaches

2.2 Additional Roles of Ontologies

Some approaches use ontologies not only for content explication, but also either as a global query model or for the verification of the (user-defined or system-generated) integration description. In the following, these additional roles of ontologies are considered in more detail.

Query Model Integrated information sources normally provide an integrated global view. Some integration approaches use the ontology as the global query schema. For example, in SIMS [Arens *et al.*, 1996] the user formulates a query in terms of the ontology. Then SIMS reformulates the global query into sub-queries for each appropriate source, collects and combines the query results, and returns the results.

Using an ontology as a query model has the advantage that the structure of the query model should be more intuitive for the user because it corresponds more to the user's appreciation of the domain. But from a database point of view this ontology only acts as a global query schema. If a user formulates a query, he has to know the structure and the contents

of the ontology; he cannot formulate the query according to a schema he would prefer personally. Therefore, it is questionable where the global ontology is an appropriate query model.

Verification During the integration process several mappings must be specified from a global schema to the local source schema. The correctness of such mappings can be considered ably improved if these can be verified automatically. A sub-query is correct with respect to a global query if the local sub-query provides a part of the queried answers, i.e. the sub-queries must be contained in the global query (query containment) [Calvanese *et al.*, 2001; Goasdoué *et al.*, 1999]. Since an ontology contains a (complete) specification of the conceptualization, the mappings can be validated with respect to the ontologies. Query containment means that the ontology concepts corresponding to the local sub-queries are contained in the ontology concepts related to the global query.

In DWQ [Calvanese *et al.*, 2001] each source is assumed to be a collection of relational tables. Each table is described in terms of its ontology with the help of conjunctive queries. A global query and the decomposed sub-queries can be unfolded to their ontology concepts. The sub-queries are correct, i.e. are contained in the global query, if their ontology concepts are subsumed by the global ontology concepts. The PICSEL project [Goasdoué *et al.*, 1999] can also verify the mapping but in contrast to DWQ it can also generate mapping hypotheses automatically which are validated with respect to a global ontology.

The quality of the verification task strongly depends on the completeness of an ontology. If the ontology is incomplete, the verification result can erroneously imagine a correct query subsumption. Since in general the completeness can not be measured, it is impossible to make any statements about the quality of the verification.

3 Ontology Representations

A question that arises from the use of ontologies for different purposes in the context of information integration is about the nature of the ontologies used. Investigating this question we mainly focus on the kind of languages used and the general structures found. We do not discuss ontology contents, because we think that the contents strongly depends on the kind of information that has to be integrated. We further restrict the evaluation to an object-centered knowledge representation system which in most systems forms the core of the languages used.

The first thing we have to notice when we investigate different approaches to intelligent information integration based on ontologies is the overwhelming dominance of systems using some variants of description logics in order to represent ontologies. The most cited language is CLASSIC [Borgida *et al.*, 1989] which is used by different systems including OBSERVER [Mena *et al.*, 1996], SIMS [Arens *et al.*, 1996] and the work of Kashyap and Sheth [Kashyap and Sheth, 1996b]. Other terminological languages used are GRAIL [Rector *et al.*, 1997] (the Tambis Approach [Stevens *et al.*, 2000]), LOOM [MacGregor, 1991] and OIL [Fensel *et al.*, 2000] which is used for terminology integration in the BUSTER approach [Stuckenschmidt and Wache, 2000].

Beside the purely terminological languages mentioned above there are also approaches using extensions of description logics which include rule bases. Known uses of extended languages are in the PICSEL system using CARIN, a description logic extended with function-free horn rules [Goasdoué *et al.*, 1999] and the DWQ [Calvanese *et al.*, 2001] project. In the latter approach $\mathcal{AL} - log$ a combination of a simple description logics with Datalog is used [Donini *et al.*, 1998]. [Calvanese *et al.*, 2001] use the Logic $\mathcal{DLR}$ a description logic with n-ary relations for information integration in the same project. The integration of description logics with rule-based reasoning makes it necessary to restrict the expressive power of the terminological part of the language in order to remain decidable [Levy and Rousset, 1996].

The second main group of languages used in ontology-based information integration systems are classical frame-based representation languages. Examples for such systems are COIN [Goh, 1997], KRAFT [Preece *et al.*, 1999], Infosleuth [Woelk and Tomlinson, 1994] and Infomaster [Genesereth *et al.*, 1997]. Languages mentioned are Ontolingua [Gruber, 1993] and OKBC [Chaudhri *et al.*, 1998]. There are also approaches that directly use F-Logic [Kifer *et al.*, 1995] with a self-defined syntax (Ontobroker [Fensel *et al.*, 1998] and COIN [Goh, 1997]). For an analysis of the expressive power of these languages, we refer to Corcho and Gomez-Perez [Corcho and Gómez-Pérez, 2000] who evaluated different ontology languages including the ones mentioned above.

4 Use of Mappings

The task of integrating heterogeneous information sources put ontologies in context. They cannot be perceived as stand-alone models of the world but should rather be seen as the glue that puts together information of various kinds. Consequently, the relation of an ontology to its environment plays an essential role in information integration. We use the term mappings to refer to the connection of an ontology to other parts of the application system. In the following, we discuss the two most important uses of mappings required for information integration: mappings between ontologies and the information they describe and mappings between different ontologies used in a system.

4.1 Connection to Information Sources

The first and most obvious application of mappings is to relate the ontologies to the actual contents of an information source. Ontologies may relate to the database scheme but also to single terms used in the database. Regardless of this distinction, we can observe different general approaches used to establish a connection between ontologies and information sources. We briefly discuss these general approaches in the sequel.

Structure Resemblance A straightforward approach to connecting the ontology with the database scheme is to simply produce a one-to-one copy of the structure of the database and encode it in a language that makes automated reasoning

possible. The integration is then performed on the copy of the model and can easily be tracked back to the original data. This approach is implemented in the SIMS mediator [Arens *et al.*, 1996] and also by the TSIMMIS system [Chawathe *et al.*, 1994].

Definition of Terms In order to make the semantics of terms in a database schema clear it is not sufficient to produce a copy of the schema. There are approaches such as BUSTER [Stuckenschmidt and Wache, 2000] that use the ontology to further define terms from the database or the database scheme. These definitions do not correspond to the structure of the database, these are only linked to the information by the term that is defined. The definition itself can consist of a set of rules defining the term. However, in most cases terms are described by concept definitions.

Structure Enrichment is the most common approach to relating ontologies to information sources. It combines the two previously mentioned approaches. A logical model is built that resembles the structure of the information source and contains additional definitions of concepts. A detailed discussion of this kind of mapping is given in [Kashyap and Sheth, 1996a]. Systems that use structure enrichment for information integration are OBSERVER [Mena *et al.*, 1996], KRAFT [Preece *et al.*, 1999], PICSEL [Goasdoué *et al.*, 1999] and DWQ [Calvanese *et al.*, 2001]. While OBSERVER uses description logics for both structure resemblance and additional definitions, PICSEL and DWQ defines the structure of the information by (typed) horn rules. Additional definitions of concepts mentioned in these rules are done by a description logic model. KRAFT does not commit to a specific definition scheme.

Meta-Annotation A rather new approach is the use of meta annotations that add semantic information to an information source. This approach is becoming prominent with the need to integrate information present in the World Wide Web where annotation is a natural way of adding semantics. Approaches which are developed to be used on the World Wide Web are Ontobroker [Fensel *et al.*, 1998] and SHOE [Heflin and Hendler, 2000b]. We can further distinguish between annotations resembling parts of the real information and approaches avoiding redundancy. SHOE is an example for the former, Ontobroker for the latter case.

4.2 Inter-Ontology Mapping

Many of the existing information integration systems such as [Mena *et al.*, 1996] or [Preece *et al.*, 1999] use more than one ontology to describe the information. The problem of mapping different ontologies is a well known problem in knowledge engineering. We will not try to review all research that is conducted in this area. We rather discuss general approaches that are used in information integration systems.

Defined Mappings A common approach to the ontology mapping problem is to provide the possibility to define mappings. This approach is taken in KRAFT [Preece *et al.*, 1999], where translations between different ontologies are done by special mediator agents which can be customized to translate between different ontologies and even different languages. Different kinds of mappings are distinguished in this approach starting from simple one-to-one mappings between classes and values up to mappings between compound expressions. This approach allows a great flexibility, but it fails to ensure a preservation of semantics: the user is free to define arbitrary mappings even if they do not make sense or produce conflicts.

Lexical Relations An attempt to provide at least intuitive semantics for mappings between concepts in different ontologies is made in the OBSERVER system [Mena *et al.*, 1996]. The approaches extend a common description logic model by quantified inter-ontology relationships borrowed from linguistics. In OBSERVER, relationships used are *synonym, hypernym, hyponym, overlap, covering* and *disjoint*. While these relations are similar to constructs used in description logics they do not have a formal semantics. Consequently, the subsumption algorithm is rather heuristic than formally grounded.

Top-Level Grounding In order to avoid a loss of semantics, one has to stay inside the formal representation language when defining mappings between different ontologies (e.g. DWQ [Calvanese *et al.*, 2001]). A straightforward way to stay inside the formalism is to relate all ontologies used to a single top-level ontology. This can be done by inheriting concepts from a common top-level ontology. This approach can be used to resolve conflicts and ambiguities (compare [Heflin and Hendler, 2000b]). While this approach allows to establish connections between concepts from different ontologies in terms of common superclasses, it does not establish a direct correspondence. This might lead to problems when exact matches are required.

Semantic Correspondences An approach that tries to overcome the ambiguity that arises from an indirect mapping of concepts via a top-level grounding is the attempt to identify well-founded semantic correspondences between concepts from different ontologies. In order to avoid arbitrary mappings between concepts, these approaches have to rely on a common vocabulary for defining concepts across different ontologies. Wache [1999] uses semantic labels in order to compute correspondences between database fields. Stuckenschmidt et. al. build a description logic model of terms from different information sources and shows that subsumption reasoning can be used to establish relations between different terminologies. Approaches using formal concept analysis (see above) also fall into this category, because they define concepts on the basis of a common vocabulary to compute a common concept lattice.

5 Ontological Engineering

The previous sections provided information about the use and importance of ontologies. Hence, it is crucial to support the

development process of ontologies. In this section, we will describe how the systems provide support for the ontological engineering process. This section is divided into three subsections: In the first subsection we give a brief overview of development methodology. The second subsection is an overview of supporting tools and the last subsection describes what happens when ontologies change.

5.1 Development Methodology

Lately, several publications about ontological developments have been published. Jones et al. [1998] provide an excellent but short overview of existing approaches (e.g. METHONTODOLOGY [Gómez-Pérez, 1998] or TOVE [Fox and Grüninger, 1998]). Uschold and Grüninger [1996] and Gómez-Pérez et al. [1996] propose methods with phases that are independent of the domain of the ontology. These methods are of good standards and can be used for comparisons. In this section, we focus on the proposed method from Uschold and Grüninger as a 'thread' and discuss how the integrated systems evaluated in this paper are related to this approach.

Uschold and Grüninger defined four main phases:

1. Identifying a purpose and scope: Specialization, intended use, scenarios, set of terms including characteristics and granularity
2. Building the ontology
   (a) Ontology capture: Knowledge acquisition, a phase interacting with requirements of phase 1.
   (b) Ontology coding: Structuring of the domain knowledge in a conceptual model.
   (c) Integrating existing ontologies: Reuse of existing ontologies to speed up the development process of ontologies in the future.
3. Evaluation: Verification and Validation.
4. Guidelines for each phase.

In the following paragraphs we describe integration systems and their methods for building an ontology. Further, we discuss systems without an explicit method where the user is only provided with information in the direction in question. The second type of systems can be distinguished from others without any information about a methodology. This is due to the fact that they assume that ontologies already exist.

Infosleuth: This system semi-automatically constructs ontologies from textual databases [Hwang, 1999]. The methodology is as follows: first, human experts provide a small number of *seed words* to represent high-level concepts. This can be seen as the identification of purpose and scope (phase 1). The system then processes the incoming documents, extracting phrases that involve seed words, generates corresponding concept terms, and then classifies them into the ontology. This can be seen as ontology capturing and part of coding (phases 2a and 2b). During this process the system also collects seed word-candidates for the next round of processing. This iteration can be completed for a predefined number of rounds. A human expert verifies the classification after each round (phase 3). As more documents arrive, the ontology expands and the expert is confronted with the new concepts. This is a significant feature of this system. Hwang calls this 'discover-and-alert' and indicates that this is a new feature of his methodology. This method is conceptually simple and allows effective implementation. Prototype implementations have also shown that the method works well. However, problems arise within the classification of concepts and distinguishing between concepts and non-concepts.

Infosleuth requires an expert for the evaluation process. When we consider that experts are rare and their time is costly this procedure is too expert-dependent. Furthermore, the integration of existing ontologies is not mentioned. However, an automatic verification of this model by a reasoner would be worthwhile considering.

KRAFT: offers two methods for building ontologies: the building of shared ontologies [Jones, 1998] and extracting of source ontologies [Pazzaglia and Embury, 1998].

Shared ontologies: The steps of the development of shared ontologies are *(a) ontology scoping, (b) domain analysis, (c) ontology formalization, (d) top-level-ontology*. The minimal scope is a set of terms that is necessary to support the communication within the KRAFT network. The domain analysis is based on the idea that changes within ontologies are inevitable and the means to handle changes should be provided. The authors pursue a domain-led strategy [Paton *et al.*, 1991], where the shared ontology fully characterizes the area of knowledge in which the problem is situated. Within the ontology formalization phase the fully characterized knowledge is defined formally in classes, relations and functions. The top-level-ontology is needed to introduce predefined terms/primitives.

If we compare this to the method of Uschold and Grüninger we can conclude that ontology scoping is weakly linked to phase 1. It appears that ontology scoping is a set of terms fundamental for the communication within the network and therefore can be seen as a vocabulary. On the other hand, the authors say that this is a *minimal* set of terms which implies that more terms exist. The domain analysis refers to phases 1 and 2a whereas the ontology formalization refers to phase 2b. Existing ontologies are not considered.

Extracting ontologies: Pazzaglia and Embury [1998] introduce a bottom-up approach to extract an ontology from existing shared ontologies. This extraction process consists of two steps. The first step is a syntactic translation from the KRAFT exportable view (in a native language) of the resource into the KRAFT-schema. The second step is the ontological upgrade, a semi-automatic translation plus knowledge-based enhancement, where local ontology adds knowledge and further relationships between the entities in the translated schema.

This approach can be compared to phase 2c, the integration of existing ontologies. In general, the KRAFT methodology lacks the evaluation of ontologies and the general purpose scope.

Ontobroker: The authors provide information about phase 2, especially 2a and 2b. They distinguish between three

classes of web information sources (see also [Ashish and Knoblock, 1997]): *(a) Multiple-instance sources* with the same structure but different contents, *(b) single-instance sources* with large amount of data in a structured format, and *(c) loosely structured pages* with little or no structure. Ontobroker [Decker *et al.*, 1999] has two ways of formalizing knowledge (this refers to phase 2b). First, sources from (a) and (b) allow to implement wrappers that automatically extract factual knowledge from these sources. Second, sources with little or no knowledge have to be formalized manually. A supporting tool called OntoEdit [Staab *et al.*, 2000] is an ontology editor embedded in the ontology server and can help to annotate the knowledge. OntoEdit is described later in this section.

Apart from the connection to phase 2 the Ontobroker system provides no information about the scope, the integration of existing ontologies, or the evaluation.

SIMS: An independent model of each information source must be described for this system, along with a domain model that must be be defined to describe objects and actions [Arens *et al.*, 1993]. SIMS model of the application domain includes a hierarchical terminological knowledge base with nodes representing objects, actions, and states. In addition, it includes indications of all relationships between the nodes. Further, the authors address the scalability and maintenance problems when a new information source is added or the domain knowledge changes. As every information source is independent and modeled separately, the addition of a new source should be relatively straightforward. A graphical LOOM knowledge base builder (LOOM-KB) can be used to support this process. The domain model would have to be enlarged to accommodate new information sources or simply new knowledge (see also [MacGregor, 1990], [MacGregor, 1988]).

The SIMS model has no concrete methodology for building ontologies. However, we see links referring to phase 2a ontology capture (description of the independent model of information sources) and 2b ontology coding (LOOM-KB). The integration of existing ontologies and an evaluation phase are not mentioned.

All the other systems discussed, such as Picsel, Observer, the approach from Kayshap & Sheth, BUSTER and COIN either have no methods or do not discuss them to create ontologies. After reading papers about these various systems it becomes obvious that there is a lack of a 'real' methodology for the development of ontologies. We believe that the systematic development of the ontology is extremely important and therefore the tools supporting this process become even more significant.

5.2 Supporting tools

Some of the systems we discussed in this paper provide support with the annotation process of sources. This process is mainly a semantic enrichment of the information therein. In the following, we sketch the currently available tools.

- OntoEdit: This tool makes it possible to inspect, browse, codify and modify ontologies and to use these features to support the ontology development and maintenance task [Staab and Mädche, 2000]. Currently, OntoEdit supports the representation languages *(a) F-Logic including an inference engine*, *(b) OIL*, *(c) Karlsruhe RDF(S)extension*, and *(d) internal XML-based serialization of the ontology model using OXML*.
- SHOE's Knowledge Annotator: With the help of this tool, the user can describe the contents of a web page [Heflin and Hendler, 2000b]. The Knowledge Annotator has an interface which displays instances, ontologies, and claims (documents collected). The tool also provides integrity checks. With a second tool called Exposé the annotated web pages can be parsed and the contents will be stored in a repository. This SHOE-knowledge is then stored in a Parka knowledge base [Stoffel *et al.*, 1997].
- DWQ: Further development within the DWQ project leads to a tool called i·com [Franconi and Ng, 2000]. i·com is a supporting tool for the conceptual design phase. This tool uses an extended entity relationship conceptual (EER) data model and enriches it with aggregations and inter-schema constraints. i·com does not provide a methodology nor is it an annotation tool, it serves mainly for intelligent conceptual modelling.

Annotation tools such as OntoEdit and the Knowledge Annotator are relatively new on the market. Therefore, comprehensive tests to give a good evaluation have yet to be done. However, we did the first steps with OntoEdit and came to the conclusion that OntoEdit seems to be a powerful tool and worthwhile considering. This is especially true when using an integration system which does not support the development process of an ontology. Also, OntoEdit allows to verify an ontology. Tests with the Knowledge Annotator have yet to be done.

5.3 Ontology Evolution

Almost every author describes the evolution of an ontology as a very important task. An integration system — and the ontologies — must support adding and/or removing sources and must be robust to changes in the information source. However, integration systems which take this into account are rare. To our knowledge, SHOE is the only system that accomplishes this to-date.

SHOE: Once the SHOE-annotated web pages are uploaded on the web, the Exposé tool has the task to update the repositories with the knowledge from these pages. This includes a list of pages to be visited and an identification of all hypertext links, category instances, and relation arguments within the page. The tool then stores the new information in the PARKA knowledge base. Heflin and Hendler [2000a] analyzed the problems associated with managing dynamic ontologies through the web. By adding revision marks to the ontology, changes and revision become possible. The authors illustrated that revisions which add categories and relations will have no effect, and that revisions which modify rules may change the answers to queries. When categories and relations are removed, answers to queries may be eliminated.

In summary, most of the authors mention the importance of a method for building ontologies. However, only few systems really support the user with a genuine method. Infosleuth is the only system which fulfills the requirements of a methodology. However, the majority of the systems only provide support of the formalization phase (please refer to phases 2a and 2b). KRAFT, SIMS, DWQ, and SHOE are representatives of this group. The remaining systems do not include a methodology. Some systems offer some support for the annotation of information sources (e.g. SHOE). Other systems provide supporting tools for parts of ontology engineering (e.g. DWQ/i·com, OntoEdit). Only the SHOE system may be considered as a system which takes ontology evolution into account.

6 Summary

In this paper we presented the results of an analysis of existing information integration systems from an ontology point of view. The analysis was focused on systems and approaches with ontologies as a main element. Important questions covered in the analysis are:

Role of the ontology: What is the purpose of the ontology and how does it relate to other parts of the systems?

Ontology Representation: What are the features (expressiveness, reasoning capabilities) of the language used to represent the ontology?

Use of Mappings: How is the connection of an ontology to other parts of the system especially data-repositories and other ontologies implemented?

Ontology Engineering: Does the approach contain a methodology and tools that support the development and the use of the ontology?

We evaluated different approaches with respect to these questions. At this point, we try to summarize the lessons learned from the analysis by drawing a rough picture of the state-of-the-art implied by the systems we analyzed. On the other hand, we try to infer open problems and to define research questions that have been put forward but reqire further investigation.

State of the Research

We tried to illustrate the state of the art by describing a 'typical' information integration system that uses well-established technologies: The typical information integration system uses ontologies to explicate the contents of an information source, mainly by describing the intended meaning of table and datafield names. For this purpose, each information source is supplemented by an ontology which resembles and extends the structure of the information source. In a typical system, integration is done at the ontology level using either a common ontology all source ontologies are related to or fixed mappings between different ontologies. The ontology language of the typical system is based on description logics and subsumption reasoning is used in order to compute relations between different information sources and sometimes to validate the result of an integration. The process of building and using ontologies in the typical system is supported by specialized tools in terms of editors.

Open Questions

The description of the typical integration system shows that reasonable results have been achieved on the technical side of using ontologies for intelligent information integration. Only the use of mappings is an exception. It seems that most approaches still use ad-hoc or arbitrary mappings especially for the connection of different ontologies. There are approaches that try to provide well-founded mappings, but they either rely on assumptions that cannot always be guaranteed or they face technical problems. We conclude that there is a need to investigate mappings on a theoretical and an empirical basis.

Beside the mapping problem, we found a striking lack of sophisticated methodologies supporting the development and use of ontologies. Most systems only provide tools. If there is a methodology it often only covers the development of ontologies for a specific purpose which is prescribed by the integration system. The comparison of different approaches, however, revealed that requirements concerning ontology language and structure depends on the kind of information to be integrated and the intended use of the ontology. We therefore think that there is a need to develop a more general methodology that includes an analysis of the integration task and supports the process of defining the role of ontologies with respect to these requirements. We think that such a methodology has to be language-independent, because the language should be selected based on the requirements of the application and not the other way round. A good methodology also has to cover the evaluation and verification of the decisions made with respect to language and structure of the ontology. The development of such a methodology will be a major step in the work on ontology-based information integration because it will help to integrate results already achieved on the technical side and to put these techniques to work in real-life applications.

References

[Arens *et al.*, 1993] Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[Arens *et al.*, 1996] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the sims information mediator. In *Advanced Planning Technology*. AAAI Press, California, USA, 1996.

[Ashish and Knoblock, 1997] Naveen Ashish and Craig A. Knoblock. Semi-automatic wrapper generation for internet information sources. In *Second IFCIS International Conference on Cooperative Information Systems*, Kiawah Island, SC, 1997.

[Belkin and Croft, 1992] N.J. Belkin and B.W. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[Borgida *et al.*, 1989] A. Borgida, Brachman a R. J., D. L. McGuiness, and L. A. Resnick. Classic: A structural data

model for objects. In *ACM SIGMOID International Conference on Management of Data*, Portland, Oregon, USA, 1989.

[Calvanese *et al.*, 2001] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Description logics for information integration. In *Computational Logic: From Logic Programming into the Future (In honour of Bob Kowalski)*, Lecture Notes in Computer Science. Springer-Verlag, 2001. To appear.

[Chaudhri *et al.*, 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. Open knowledge base connectivity (okbc) specification document 2.0.3. Technical report, SRI International and Stanford University (KSL), April 1998.

[Chawathe *et al.*, 1994] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous information sources. In *Conference of the Information Processing Society Japan*, pages 7–18, 1994.

[Corcho and Gómez-Pérez, 2000] Oscar Corcho and Asuncion Gómez-Pérez. Evaluating knowledge representation and reasoning capabilities of ontology specification languages. In *Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, Berlin, 2000.

[Decker *et al.*, 1999] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editor, *Semantic Issues in Multimedia Systems. Proceedings of DS-8*, pages 351–369. Kluwer Academic Publisher, Boston, 1999.

[Donini *et al.*, 1998] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Al-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems (JIIS)*, 27(1), 1998.

[Fensel *et al.*, 1998] Dieter Fensel, Stefan Decker, M. Erdmann, and Rudi Studer. Ontobroker: The very high idea. In *11. International Flairs Conference (FLAIRS-98)*, Sanibal Island, USA, 1998.

[Fensel *et al.*, 2000] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. Oil in a nutshell. In *12th International Conference on Knowledge Engineering and Knowledge Management EKAW 2000*, Juanles-Pins, France, 2000.

[Fox and Grüninger, 1998] Mark S. Fox and Michael Grüninger. Enterprise modelling, fall 1998, pp. 109-121. *AI Magazine*, 19(3):109–121, 1998.

[Franconi and Ng, 2000] Enrico Franconi and Gary Ng. The i.com tool for intelligent conceptual modelling. In *7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00)*, Berlin, Germany, August 2000, 2000.

[Genesereth *et al.*, 1997] Michael R. Genesereth, Arthur M. Keller, and Oliver Duschka. Infomaster: An information integration system. In *1997 ACM SIGMOD Conference.*, 1997.

[Gómez-Pérez *et al.*, 1996] Ascuncion Gómez-Pérez, M. Fernández, and A. de Vicente. Towards a method to conceptualize domain ontologies. In *Workshop on Ontological Engineering, ECAI '96*, pages 41–52, Budapest, Hungary, 1996.

[Gómez-Pérez, 1998] A. Gómez-Pérez. Knowledge sharing and reuse. In Liebowitz, editor, *The handbook on Applied Expert Systems*. ED CRC Press, 1998.

[Goasdoué *et al.*, 1999] François Goasdoué, Véronique Lattes, and Marie-Christine Rousset. The use of carin language and algorithms for information integration: The picsel project,. *International Journal of Cooperative Information Systems (IJCIS)*, 9(4):383 – 401, 1999.

[Goh, 1997] Cheng Hian Goh. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources*. Phd, MIT, 1997.

[Gruber, 1993] Tom Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gruber, 1995] Tom Gruber. Toward principles for the design of ontologies used for knowledge sharing, 1995.

[Heflin and Hendler, 2000a] Jeff Heflin and James Hendler. Dynamic ontologies on the web. In *Proceedings of American Association for Artificial Intelligence Conference (AAAI-2000)*, Menlo Park, CA, 2000. AAAI Press.

[Heflin and Hendler, 2000b] Jeff Heflin and James Hendler. Semantic interoperability on the web. In *Extreme Markup Languages 2000*, 2000.

[Hwang, 1999] Chung Hee Hwang. Incompletely and imprecisely speaking: Using dynamic ontologies for representing and retrieving information. Technical, Microelectronics and Computer Technology Corporation (MCC), June 1999.

[Jones *et al.*, 1998] D. M. Jones, T.J.M. Bench-Capon, and P.R.S. Visser. Methodologies for ontology development. In *Proc. IT&KNOWS Conference of the 15th IFIP World Computer Congress*, Budapest, 1998. Chapman-Hall.

[Jones, 1998] D.M. Jones. Developing shared ontologies in multi agent systems. Tutorial, 1998.

[Kashyap and Sheth, 1996a] V. Kashyap and A. Sheth. Schematic and semantic semilarities between database objects: A context-based approach. *The International Journal on Very Large Data Bases*, 5(4):276–304, 1996.

[Kashyap and Sheth, 1996b] Vipul Kashyap and Amit Sheth. Semantic heterogeneity in global information systems: The role of metadata, context and ontologies. In M. Papazoglou and G. Schlageter, editors, *Cooperative Information Systems: Current Trends and Applications*. 1996.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based systems. *Journal of the ACM*, 1995.

[Kim and Seo, 1991] Won Kim and Jungyun Seo. Classifying schematic and data heterogeinity in multidatabase systems. *IEEE Computer*, 24(12):12–18, 1991. problem classification of semantic heterogeneity.

[Levy and Rousset, 1996] Alon Y. Levy and Marie-Christine Rousset. Carin: A representation language combining horn rules and description logics. In *Proceedings of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pages 323–327, 1996.

[MacGregor, 1988] Robert M. MacGregor. A deductive pattern matcher. In *Seventh National Conference on Artificial Intelligence, (AAAI 88)*, pages 403–408, 1988.

[MacGregor, 1990] Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufman, 1990.

[MacGregor, 1991] Robert M. MacGregor. Using a description classifier to enhance deductive inference. In *Proceedings Seventh IEEE Conference on AI Applications*, pages 141–147, 1991.

[Mena *et al.*, 1996] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings 1st IFCIS International Conference on Cooperative Information Systems (CoopIS '96)*. Brussels, 1996.

[Paton *et al.*, 1991] R.C Paton, H.S. Nwana, M.J.R. Shave, T.J.M. Bench-Capon, and S. Hughes. Foundations of a structured approach to characterising domain knowledge. *Cognitive Systems*, 3(2):139–161, 1991.

[Pazzaglia and Embury, 1998] J-C.R. Pazzaglia and S.M. Embury. Bottom-up integration of ontologies in a database context. In *KRDB'98 Workshop on Innovative Application Programming and Query Interfaces*, Seattle, WA, USA, 1998.

[Preece *et al.*, 1999] A.D. Preece, K.-J. Hui, W.A. Gray, P. Marti, T.J.M. Bench-Capon, D.M. Jones, and Z. Cui. The kraft architecture for knowledge fusion and transformation. In *Proceedings of the 19th SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence (ES'99)*. Springer, 1999.

[Rector *et al.*, 1997] A.L. Rector, S. Bechofer, C.A. Goble, I. Horrocks, W.A. Nowlan, and W.D. Solomon. The grail concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139 – 171, 1997.

[Staab and Mädche, 2000] S. Staab and A. Mädche. Ontology engineering beyond the modeling of concepts and relations. In *ECAI'2000 Workshop on on Applications of Ontologies and Problem-Solving Methods*, Berlin, 2000.

[Staab *et al.*, 2000] S. Staab, M. Erdmann, and A. Mädche. An extensible approach for modeling ontologies in rdf(s). In *First ECDL'2000 Semantic Web Workshop*, Lisbon, Portugal, 2000.

[Stevens *et al.*, 2000] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N.W. Paton, C.A. Goble, and A. Brass. Tambis: Transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184–186, 2000.

[Stoffel *et al.*, 1997] Kilian Stoffel, Merwyn Taylor, and James Hendler. Efficient management of very large ontologies. In *American Association for Artificial Intelligence Conference (AAAI-97)*, pages 442–447, Menlo Park, CA, 1997. AAAI/MIT Press.

[Stuckenschmidt and Wache, 2000] Heiner Stuckenschmidt and Holger Wache. Context modelling and transformation for semantic interoperability. In *Knowledge Representation Meets Databases (KRDB 2000)*. 2000.

[Stuckenschmidt *et al.*, 2000a] H. Stuckenschmidt, Frank van Harmelen, Dieter Fensel, Michel Klein, and Ian Horrocks. Catalogue integration: A case study in ontology-based semantic translation. Technical Report IR-474, Computer Science Department, Vrije Universiteit Amsterdam, 2000.

[Stuckenschmidt *et al.*, 2000b] Heiner Stuckenschmidt, Holger Wache, Thomas Vögele, and Ubbo Visser. Enabling technologies for interoperability. In Ubbo Visser and Hardy Pundt, editors, *Workshop on the 14th International Symposium of Computer Science for Environmental Protection*, pages 35–46, Bonn, Germany, 2000. TZI, University of Bremen.

[Uschold and Grüniger, 1996] M. Uschold and M. Grüniger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[Uschold and Grüninger, 1996] Mike Uschold and Michael Grüninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[Wache *et al.*, 1999] H. Wache, Th. Scholz, H. Stieghahn, and B. König-Ries. An integration method for the specification of rule–oriented mediators. In Yahiko Kambayashi and Hiroki Takakura, editors, *Proceedings of the International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 109–112, Kyoto, Japan, November, 28-30 1999.

[Wache, 1999] Holger Wache. Towards rule-based context transformation in mediators. In S. Conrad, W. Hasselbring, and G. Saake, editors, *International Workshop on Engineering Federated Information Systems (EFIS 99)*, Kühlungsborn, Germany, 1999. Infix-Verlag.

[Woelk and Tomlinson, 1994] Darrell Woelk and Christine Tomlinson. The infosleuth project:intelligent search management via semantic agents. In *Second World Wide Web Conference '94: Mosaic and the Web*, 1994.

Short Papers and Statements

Experience in Ontology Engineering for a Multi-Agents Corporate Memory System

Fabien Gandon
ACACIA project - INRIA, 2004, route des Lucioles, B.P. 93
06902 Sophia Antipolis, France - Fabien.Gandon@sophia.inria.fr

Abstract

XML and multi-agents technologies offer a number of assets for corporate memory management. Since ontologies appear as a key asset in the new generation of information systems and also in the communication layer of multi-agents systems, it comes with no surprise that it stands out as a keystone of multi-agents information systems. Here, we briefly describe our approach and motivations and then focus on the first elements of our return on experience in building an ontology for such a system.

1 Introduction

In the last decade information systems became backbones of organizations and the industrial interest in methodologies and tools enabling capitalization and management of corporate knowledge grew stronger. A corporate memory is an explicit, disembodied and persistent representation of knowledge and information in an organization, in order to facilitate their access and reuse by members of the organization, for their tasks [Rabarijaona *et al.*, 2000]. The stake in building a corporate memory management system is the coherent integration of this dispersed knowledge in a corporation with the objective to "promote knowledge growth, promote knowledge communication and in general preserve knowledge within an organization" [Steels, 1993]. ACACIA, our research team, is part of the CoMMA project (IST-1999-12217) funded by the European Commission, aiming at implementing a corporate memory management framework based on several emerging technologies: agents, ontologies, XML, information retrieval and machine learning techniques [CoMMA, 2000]. These technical choices are mainly motivated by three observations. **(1) The memory is, by nature, an heterogeneous and distributed information landscape.** The corporate memories are now facing the same problem of precision and recall than the Web. The initiative of a semantic Web is a promising approach where the semantics of documents is made explicit through metadata and annotations to guide the later exploitation of these documents. XML enables us to build a structure around the data, and RDF (Resource Description Framework) allows resources to be semantically annotated. **(2) The tasks as a whole to be performed on the memory are, by nature, distributed and heterogeneous.** So we envisaged a distributed and heterogeneous system to explore and exploit this information landscape: a multi-agents system (MAS). It allows the resources to remain localized and heterogeneous while enabling to capitalize an integrated and global view of the memory thanks to cooperating software agents distributed over the network and having different skills and roles to support the memory tasks. The heterogeneity and distribution of the MAS is an answer to the heterogeneity and the distribution of the corporate memory. **(3) The population of the users of the memory is, by nature, heterogeneous and distributed in the corporation.** Agents will also be in charge of interfacing users with the system. Adaptation and customization are a keystone here and we are working on machine learning techniques in order to make agents adaptive to the users and the context. This goes from basic customization to user's habits and preferences learning, up to push technologies based on interest groups and collaborative filtering.

2 Approach Overview

Compared to the Web, a corporate memory has more delimited and defined context, infrastructure and scope ; the existence of a community of stakeholders means that an ontological commitment is conceivable to a certain extend. So far, the enterprise modeling field has been mainly concerned with simulation and optimization of the design of the corporate production system but last decade changes led enterprises to become aware of the value of their memory and the fact that enterprise models have a role to play in this application too. The corporation has its own organization and infrastructure ; this state of affair can be formally made explicit to guide the corporate memory activities involved, for instance, in the new employee integration and the technology monitoring scenarios of CoMMA. This enables the system to get insight into the organizational context and environment and to intelligently exploit it in interactions between agents and between agents and users. Likewise, the users' profile captures all aspects of the user that were identified as relevant for the system behavior. It contains administrative information and directly explicited

preferences that go from interface customization to topic interests. It also positions the user in the organization: role, location and potential acquaintance network. In addition to explicitly stated information, the system will derive information from the usage made by the user. It will collect the history of visited documents and possible feedback from the user, as well as the user's recurrent queries, failed queries, and from this it can learn some of the user's habits and preferences. These derived criterions can then be used for interface purposes or push technology. Finally the profiles enable to compare users, to cluster them based on similarities in their profiles and then use the similar profiles to make suggestions.

The figure 1 gives the OSA modeling architecture use in CoMMA. Our approach is : (1) to apply knowledge engineering techniques to provide the conceptual vocabulary needed by the scenarios and to formalize this ontology in RDF using the RDF Schema (2) to describe the organizational state of affair and users' profile in RDF statements (3) to structure the corporate memory with RDF annotations based on the ontology and referencing the state of affair (4) to use the annotations, the state of affair and the ontology through inferences in order to search, manage and navigate into the memory. As shown in figure 1, the ontology and the state of affair form the model ; the archive annotations will depend on both. The state of affair and the annotations are instances of the RDF schema : the ontology is at the intensional level whereas the state of affair and the annotations are at the extensional level. The ontology, the state of affair and the annotations are tightly linked and will evolve as a whole in a prototype life cycle style.

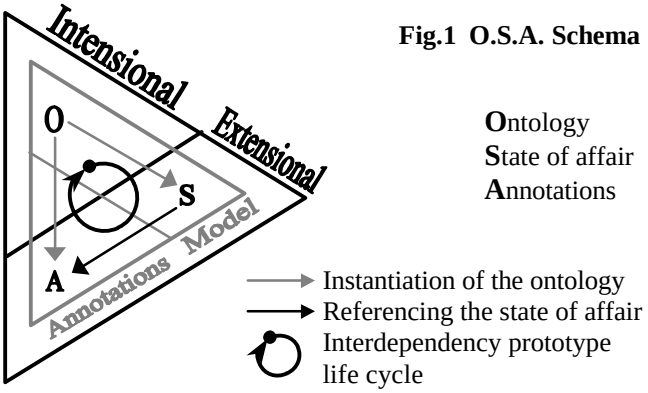


Fig.1 O.S.A. Schema

CoMMA is an heterogeneous Multi-Agents Information System (MAIS) supporting information distribution. The duality of the definition of the word 'distribution' reveals two important problems to be addressed : (a) Distribution means dispersion, that is the spatial property of being scattered about, over an area or a volume ; the problem here is to handle the naturally distributed data, information or knowledge of the organization. (b) Distribution also means the act of spreading or apportioning ; the problem then is to make the relevant pieces of information go to the concerned agent. In a MAS, distribution is handled through cooperation so in our case, agents must be able to communicate with the others to delegate tasks or solve queries. The content of the exchanged messages relies on the ontology. The agents play roles and are organized in societies as described in [Gandon *et al.*, 2000]. In order to manipulate the ontology, the annotations, and infer from them, the agents import modules from CORESE a prototype of a search engine enabling inferences on RDF annotations by translating the RDF triplets to Conceptual Graphs and vice versa [Corby et al., 2000].

3 Engineering an ontology

Following Caroll [1997] we used *scenarios* to capture end-users'needs in their context. They enable us to focus on the specific aspects of knowledge management involved in our case, to capture the whole picture and a concrete set of interaction sequences, and to view the system as a component of a knowledge management solution for a company. A scenario template was proposed, suggesting key aspects to be considered when describing a scenario and collecting data. This helps define the scope of our intervention and thus the scope of the ontology. Scenario analysis produced reports which are extremely rich story-telling documents and therefore good candidates to be included in the corpus of a terminological study.

Several techniques exist for *data collection*, we used three of them: semi-structured interview, observation and document analysis. Data collection also included the study of existing ontologies: the Enterprise Ontology [Uschold *et al.*, 1998], the TOVE Ontology [TOVE, 2000], the Upper Cyc Ontology [Cyc, 2000], the PME Ontology [Kassel *et al.*, 2000] and the CGKAT & WebKB Ontology [Martin and Eklund, 2000 ; Martin, 1996]. The reuse of ontologies is both seductive (saves time, efforts and favors standardization) and difficult (commitments and conceptualizations have to be aligned between the reused ontologies and the needed one). These ontologies have not been imported directly, the best way for us to use them was to start from their informal version in natural language. Natural language processing tools could help this analysis, and translators between formal languages could ease reuse. Reused sources have to be pruned ; scenarios capture the scope of the intervention and a shared vision of the stakeholders, they can be used to decide whether or not a concept is relevant e.g.: the 'ownership' relation of the Enterprise Ontology was not reused in our ontology because this relation does not seem exploitable in our scenarios. We also considered other informal sources: some very general ones helped us structure upper parts of some branches e.g.: the book '*Using Language*' from H.H. Clark inspired the branch on representation systems ; others very specific enabled us to save time on enumerating some leaves of the taxonomical tree e.g.: the MIME standard for electronic format description. The systematic use of dictionaries or available lexicons is good practice. In particular, the meta-dictionaries have proved to be extremely useful. They enable access to a lot of dictionaries and therefore one can easily compare definitions and identify or build the one that correspond to the notion one wants to introduce. We made extensive use of [OneLook, 2000].

The *candidate terms* were collected in a set of informal tables. The next step is to produce consensual definitions to build the concepts defined 'in intension'. At this point, labeling concepts with one term is both convenient and dangerous. It is a major source of 'ambiguity relapse' where people relapse in ambiguity using the label terms according to the definition they associate to it and not the definition actually associated to it during the semantic commitment. The explicit representation and the existence of management functionality for terminological aspects in tools assisting ontologists are real needs. The obtained concepts were organized in a taxonomy: we started regrouping concepts firstly in an intuitive way, then iteratively organizing and reviewing the structure. We studied several principles to build the taxonomical tree: the extended Aristotelian principles in [Bachimont, 2000], the semantic axis in [Kassel *et al*., 2000], and the extensive work of Guarino and Welty [Guarino, 1992; Guarino and Welty, 2000]. The main problem is that, as far as we know, no tool is available to help an ontologist apply these principles easily and independently of a formalization language; it can become a titanic work to apply these theories to large ontologies.

The way to design an ontology is still debated in the knowledge engineering community. There is a tendency to distinguish between three approaches: *Bottom-Up, Top-Down and Middle-Out*. We are not convinced that there exists such a thing as a purely top-down, bottom-up or middle-out approach. They seem to be *three complementary perspectives of a complete methodology* with concurrent processes present and at work at different levels of depth (bottom, middle or top) and different detail grains (concepts or groups of concepts). We shall not deny that for a given case, an approach can mainly rely on one perspective, but we would not oppose them as different approaches: when engineering an ontology, an ontologist should have the tasks defined in these three perspectives on the go at one time. In our case, some tasks were performed in parallel in the different perspectives, e.g. : we studied existing top-ontologies and upper parts of relevant ontologies to structure our top part and reuse parts of existing taxonomies (top-down approach); we studied different branches, domains, micro-theories of existing ontologies as well as core subjects identified during data collection to understand what were the main areas we needed and group candidate terms (middle-out approach); we exploited reports from scenario analysis and data collection traces to list scenario specific concepts and then started to regroup them by generalization (bottom-up approach). The different buds (top concepts, core concepts, specific concepts) opening out in the different perspectives are the origins of partial sub-taxonomies. The objective then is to ensure the joint of the different approaches and an event in one perspective triggers checks and tasks in others.

This approach resulted in a more or less *three-layered ontology*: (1) A very general top (2) A very large middle layer divided in two main branches: one generic for corporate memory domain and one dedicated to the topics of the application domain (3) An extension layer which tends to be scenario and company specific with internal complex concepts. We obtained three semi-informal tables (concepts, relations and attributes) with the following columns: (1) the label of the concepts / relations / attributes, (2) the concepts linked by the relation or the concept and the basic type linked by the attribute, (3) the closet core concept or the thematic fields linked by the relation, (4) the inheritance links, (5) synonymous terms for the label, (6) a natural language definition to try to capture the intension, and (7) the collection source. This last column introduces the principle of traceability and it is interesting for the purpose of abstracting a methodology from the work done. It enables to know what sort of contribution influenced a given part of the ontology and to trace the effectiveness of reuse. However this is by far not enough and the complete design rationale of the ontology should be captured in order to help people understand and may be commit to or adapt it.

The final formal degree of the ontology depends on its intended use. *The goal of the formalization task is not to take an informal ontology and translate it into a rigorously formal ontology, but to develop the formal counterpart of interesting and relevant semantic aspects of the informal ontology* in order to obtain a documented (informal description possibly augmented by navigation capabilities from the formal description) operational ontology (formal description of the relevant semantic attributes needed for the envisioned system). The formal form of an ontology must include the natural language definitions, comments, remarks, that will be exploited by humans trying to appropriate the ontology. This also plays an important role for documenting the ontology and therefore for ontology reuse, reengineering and reverse-engineering.

In our case, the last step of formalization was the *translation of semi-informal tables in RDF*. Thanks to the XML technology we managed to keep the informal view through XLST style sheets: (a) a style sheet recreates the table of concepts (b) a second one recreates the table of relations and attributes (c) a last one proposes a new view as a tree of concepts with their attached definition as a popup window following the mouse pointer. This pop-up is a first attempt to investigate how to proactively disambiguate navigation or querying: before the user clicks on a concept, the system displays the natural language definition inviting the user to check his personal definition upon the definition used by the system so as to avoid misunderstandings. The second interesting point of that view is that if the user clicks on a concept he obtains all the instances of this concept and its sub-concepts, so this view is a link between the intensional level and the extensional one.

The design of an ontology is an iterative maturation process, it follows a prototype life-cycle [Fernandez *et al*., 1997]. As an example, one of the problems spotted when reviewing the ontology was the redundancy ; for instance we found that annotating a document as multi-modal is redundant with the fact that we annotated it with the different modes it uses. So we decided that the multi-modal was not a basic annotation concept and that it should be a defined concept derived from other existing concepts where

possible. However the notion of defined concept, does not exist in RDFS, and we will have to extend the schema as proposed in [Delteil *et al.*, 2001].

The first draft of the ontology was a good step for feasibility study and first prototypes, but it comes with no surprise that the prototype life-cycle is time consuming. Moreover the ontology is a living object the maintenance of which has consequences beyond its own life-cycle : what happens to the annotations written thanks to this conceptual vocabulary when a change occurs in the ontology? Deletion and modification obviously raise the crucial problem of coherence and correction of the annotation base. But an apparently innocuous addition of a concept also raises the question of the annotations using a parent concept of the new concept and that could have been more precise if the concept had existed when they were formulated: should we review them or not ? These problems are obviously even more complex in the context of a distributed system. Finally, an ergonomic representation interface is a critical factor for the adoption of the ontology by the users; if the user is overloaded with details or lost in the meanderings of the taxonomy he will never use the system and the life-cycle of the ontology will never complete a loop. We are investigating that point, and the terminological level seems very important here too.

4 Conclusion

Ontologies are a keystone of multi-agent systems and play an important role in the new generation of information systems, therefore they will clearly become a central component of MAIS and they surely do in CoMMA. Our experience gave rise to several expectations and to be able to manage, share and discuss the growing ontology, we would definitively need an integrated environment with: (a) improved interfaces for representation, navigation and manipulation of ontologies (b) natural language processing tools to semi-automate the analysis of the extensive part of the resources that are textual (c) facilities for applying the results from theoretical foundations of Ontology and help ontologists check their ontologies (d) tools to manage the versioning of the ontology and all that has been built upon it (annotations, models, inferences...) and to capture the design rationale. Finally work is needed to help make explicit and preserve the intensional semantic structure of the computational level. If the new generation of AI agents is to be based on an explicit conceptualization, this must not be limited to the knowledge exchanged currently, it must include the action performed on it with both their intension and intention.

Acknowledgements

I warmly thank my colleagues Rose Dieng, Olivier Corby and Alain Giboin, the CoMMA consortium and the European Commission funding the CoMMA project.

References

[Bachimont, 2000] Bachimont, Engagement sémantique et engagement ontologique: conception et réalisation d'ontologies en ingénierie des connaissances, In J. Charlet et al., *Ingénierie des connaissances Evolutions récentes et nouveaux défis*, Eyrolles

[Caroll, 1997] Caroll, Scenario-Based Design, In Helander et al. *Handbook of Human-Computer Interaction.*, Chap. 17, Elsevier Science B.V.

[CoMMA, 2000] CoMMA, Corporate Memory Management through Agents, In *Proc. E-Work E-Business*

[Corby *et al.*, 2000] Corby, Dieng, Hébert. A Conceptual Graph Model for W3C Resource Description Framework. In *Proc. ICCS'2000*

[Cyc, 2000] www.cyc.com/cyc-2-1/cover.html

[Delteil *et al.*, 2001] Delteil, Faron, Dieng, Extension of RDFS based on the CG formalism, In *Proc. ICCS'01*

[Fernandez *et al.*, 1997] Fernandez, Gomez-Perez, Juristo. METHONTOLOGY: From Ontological Arts Towards Ontological Engineering. In *Proc. AAAI97 Symposium Ontological Engineering*

[Gandon *et al.*, 2000] Gandon, Dieng, Corby, Giboin, A Multi-Agents System to Support Exploiting an XML-based Corporate Memory, In *Proc. PAKM'00*

[Guarino and Welty, 2000] Guarino, Welty, Towards a methodology for ontology-based model engineering. In *Proc. ECOOP-2000 Workshop Model Engineering.*

[Guarino, 1992] Guarino, Concepts, Attributes, and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. In *Data and Knowledge Engineering* 8

[Kassel *et al.*, 2000] Kassel, Abel, Barry, Boulitreau, Irastorza, Perpette, Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche. In *Proc. IC'00*

[Martin and Eklund, 2000] Martin, Eklund, Knowledge Indexation and Retrieval and the Word Wide Web. *IEEE Intelligent Systems special issue Knowledge Management over the Internet.*

[Martin, 1996] Martin, Ph.D. Thesis Exploitation de Graphes Conceptuels et de documents structurés et Hypertextes pour l'acquisition de connaissances et la recherche d'informations, University of Nice Sophia Antipolis

[OneLook, 2000] www.onelook.com/

[Rabarijaona *et al.*, 2000] Rabarijaona, Dieng, Corby, Ouaddari, Building and searching a XML-based Corporate Memory, *IEEE Intelligent Systems Special Issue Knowledge Management and Internet* 56-64

[Steels, 1993] Steels, Corporate Knowledge Management. In Barthès ed., *Proc. ISMICK'93*

[TOVE, 2000] www.eil.utoronto.ca/tove/ontoTOC.html

[Uschold *et al.*, 1998] Uschold, King, Moralee, Zorgios, The Enterprise Ontology. In Uschold and Tate *The Knowledge Engineering Review Special Issue on Putting Ontologies to Use*, Vol. 13

Statement of Interest: Towards Ontology Language Customization

Heiner Stuckenschmidt
Center for Computing Technologies
University of Bremen

1 Motivation

It has been argued that intelligent applications benefit from the use of ontologies encoding background knowledge about the structure of a domain and the meaning of terms occurring therein. Prominent examples can be found in the following application areas:

Systems Engineering: The use of ontologies for the description of information and systems has many benefits. The ontology can be used to identify requirements as well as inconsistencies in a chosen design. It can help to acquire or search for available information. Once a systems component has been implemented its specification can be used for maintenance and extension purposes.

Information Integration: An important application area of ontologies is the integration of existing systems. The ability to exchange information at run time, also known as interoperability, is an important topic. In order to enable machines to understand each other we also have to explicate the vocabulary of each system in terms of an ontology.

Information Retrieval: Common information-retrieval techniques either rely on a specific encoding of available information (e.g. fixed classification codes) or simple full-text analysis. Both approaches suffer from severe shortcomings. Using an ontology in order to explicate the vocabulary can help overcome some of these problems. When used for the description of available information as well as for query formulation an ontology serves as a common basis for matching queries against potential results on a semantic level.

These application areas come with completely different requirements concerning the modeling and reasoning abilities of the ontology language used. In turn, existing ontology languages are rather generic, because they aim at providing modeling facilities independent of a concrete application. In principle, being generic is an advantage, because a generic language covers broader range of applications. However, the use of generic languages turns out to produce problems in real-life applications. These problems include the following:

- *Natural distinctions of an application domain are not supported by the language:* In the design of knowledge-based systems the distinction between tasks and methods (and their ontologies) is an important one.
- *The use of modeling constructs in a concrete application is not obvious:* A re-occurring discussion in ontological modeling is whether to represent a domain item as a class or an instance.
- *Small languages are not used, because implicitly representable constructs are overlooked:* The concept of disjointness of a set of classes can be modeled by negation and Implication.
- *Unused language features lead to unnecessary complexity of the language:* Transitive slots are a modeling construct not used too often that is hard to handle with respect to inference.

We argue that ontology language can gain practical relevance if they would address these problems. A promising way is to provide a framework that allows to customize an ontology language with respect to a given application. A customized language should cover the natural distinctions of a domain and provide guidance for the use of language constructs. Further it should be designed as an optimal trade off between reasoning expressiveness and reasoning complexity.

2 The Representation-Reasoning Trade-Off of Ontology Languages

We exemplify the representation-reasoning trade-off of ontology languages and its impact on the application of the language using ontology languages that are based on description logics. The rational for this choice is:

- The expressiveness and complexity of these languages has been studied thoroughly and well-founded results are available [Donini *et al.*, 1991]
- It has been shown that description logics provide a unifying framework for many class-based representation formalisms [Calvanese *et al.*, 1999].
- Description logic-based languages have become of interest in connection with the semantic web. the languages OIL [Fensel *et al.*, 2000] and DAML-ONT [McGuinness *et al.*, 2000] are good examples.

We compared three description logic languages that have been used to build ontologies, i.e. CLASSIC, LOOM and OIL. The results of the comparison are depicted in figure 1.

	CLASSIC	OIL	LOOM
Logical Operators			
conjunction	×	×	×
disjunction		×	×
negation		×	×
Slot-Constraints			
slot values	×		×
type restriction	×	×	×
range restriction	×	×	×
existential restriction	×	×	×
cardinalities	×	×	×
Assertions			
entities	×	(×)	×
relation-instances	×	(×)	×

Figure 1: Expressiveness of some description logic based ontology languages

The comparison reveals an emphasis on highly expressive concept definitions. The languages compared are capable of almost all common concept forming operators. An exception is CLASSIC that does not allow the use of disjunction and negation in concept definitions. The reason for this shortcoming is the existence of a sound and complete subsumption algorithm that support A-box reasoning [Borgida and Patel-Schneider, 1994]. LOOM on the other hand is a very expressive language containing all language constructs used in the comparison. The price for this high expressiveness is a loss in reasoning support: Soundness and completeness of the subsumption algorithms cannot be guaranteed [Horrocks, 1995].

The OIL approach is a first attempt to overcome the problems that arise from the representation-reasoning trade-off by defining a family of languages of different complexity. While the purpose of the smallest language of the OIL family (*Core OIL*) is to define a well-founded semantics for schemas of the emerging web standard RDF. This language is rather small and therefore allows efficient reasoning. The main language *Standard OIL* is tailored to have efficient reasoning support for consistency checking and for automatic construction of inheritance hierarchies for an extremely expressive logic. However this language does not include assertional language, because this would disable the reasoning support. For applications where instances are required, the OIL defines the language *Instance-OIL* that includes instances, but has no reasoning support.

3 The Design Space of Ontology Languages

The OIL framework allows a user to select between languages of different expressive power, however it does not address the problem of tailoring a language to a given application. Our main objective is that the current architecture of the OIL framework does only allow for strict extensions excluding the possibility to define alternative language that only partially overlap.

In order to allow more flexible variations we have to investigate the design space of ontology languages. There are many options to be taken into account. We could rely on previous work on comparing frame-based and terminological knowledge representation systems [Karp, 1993; Heinsohn *et al.*, 1994]. As our concerns are rather application driven than of a theoretical nature, we have to abstract from the technical details of the languages that are mainly concerned in the work mentioned above. We therefore concentrate on the following questions:

- What kinds of knowledge have to be modeled ?
- Which reasoning tasks have to be performed ?
- Which level of complexity is acceptable ?

The answers to these questions depend on the purpose of the language. They constitute dimensions of the design space: Different types of knowledge can be used for different kind of reasoning tasks. Further different kinds of reasoning methods result in different levels of reasoning complexity. In order to customize a language, we have to locate it with respect to each of these dimensions. Possible locations are further restricted by the needs and possibilities of the application environment. Examples for further design constraints are:

- The conceptualization of the application domain as well as pre-existing models implies the existence of certain knowledge types. The designed language must at least implicitly support these knowledge types.
- The role of the ontology in the overall application implies a certain task type. The design space is therefore restricted to variations of this task type.
- The availability of reasoners for the given task does not only have impact on the reasoning complexity, but also on the types of knowledge that can be used to define the ontology.

These restrictions have to be taken into account when the design space is explored. The situation becomes complicated, because the dimensions are not independent of each other. We already mentioned the interrelation of modeling primitives and reasoning support. In order to resolve such conflicts an engineering method is needed to guide the search process and validate the result.

4 A Pattern-Based Approach

We propose an engineering approach for customizing ontology languages that is based on the notion of ontology patterns. A pattern denote a language construct with special properties with respect to structure, semantics and inference capabilities. In a related approach Staab an colleagues propose the use of *semantic patterns* to support ontology engineering and propose a set of such patterns [Staab and Maedche, 2000]. The idea of providing a set of simple patterns that can be combined to form more complex patterns on which languages can be based is essential, because it makes different customized languages comparable and

provides a basis for translations across these languages. As already mentioned, we restrict ourselves to description-logic based languages. Therefore, our set of modeling primitives to start with are the concept forming operators of these kinds of languages.

Relying on description logics we already get a notion of more complex patterns in terms of special logics. These logics result from the combination of operators. The name of the language and therefore the pattern is a combination of the identifiers of the operators included. One of the most well known patterns is $\mathcal{ALC}$ the description logic containing Boolean operations on class expressions as well as universal and existential restrictions on slot fillers. The pattern used to resemble different class-based representation formalisms in [Calvanese *et al.*, 1999] is $\mathcal{ALUNI}$ which contains the corresponding operators: conjunction, disjunction, negation, universal restrictions on slot fillers, quantified number restrictions and inverse slots. $\mathcal{SHIQ}$, the logic underlying OIL also contains existential restrictions, transitive slots and conjunction of slot definitions. Theoretical results from the field of description logics provide us with the knowledge about decidable combinations of modeling primitives and their complexity with respect to subsumption reasoning. Consequently, every decidable combination of operators is a potential pattern that can be used to build the ontology for a certain application. In the course of the engineering process we have to handle different patterns:

Reasoner Patterns describe the language a certain reasoner is able to handle.

Reuse Patterns describe the language a useful, already existing ontology in encoded in.

Acquisition Patterns describe the language needed to encode acquired knowledge.

The Goal Pattern describes the language that will be designed.

In order to find the goal pattern, we have to find an optimal trade-off between the other patterns involved. For this purpose we invent the notion of coverage for ontology patterns. A Pattern P_1 is said to cover a pattern P_2, if all modeling primitives from P_2 are also contained in $P2_1$ or can be simulated by a combination of modeling primitives from P_1. We denote the fact that P_1 covers P_2 as $P_2 \prec P_1$. Using the notion of coverage we can now define the customization task.

Definition: Customization Task. A customization task is defined by a three tuple $\langle \mathcal{R}, \mathcal{U}, \mathcal{A} \rangle$ where $\mathcal{R}$ is a set of reasoner patterns, $\mathcal{U}$ a set of reuse patterns and $\mathcal{A}$ a set of acquisition patterns. The pattern G is a solution of the customization task if it is the minimal pattern that is covered by a reasoner pattern and covers all reuse and acquisition patterns, or formally:

Suitability of the goal pattern:

$$suitable(G) \iff \exists R \in \mathcal{R}(G \prec R) \wedge \forall P \in \mathcal{U} \cup \mathcal{A}(P \prec G)$$

Minimality of the goal pattern:

$$minimal(G) \iff \neg \exists G'(suitable(G') \wedge G' \prec G)$$

Solution:

$$solution(G) \iff suitable(G) \wedge minimal(G)$$

This definition provides us with an idea of the result of the customization process. However there are still many technical and methodological problems. We have to investigate the nature of the covering predicate and develop an algorithm for deciding whether one pattern covers the other. Further, the customization process has to be implemented. It is quite likely that the acquisition pattern will not be completely available in the beginning. Therefore we have to incorporate user interaction and revisions of previous decisions. Finally, results have to be generalized beyond the scope of description logics which will be difficult, because there are less theoretical results to build upon.

References

[Borgida and Patel-Schneider, 1994] A. Borgida and P.F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, 1(2):277–308, 1994.

[Calvanese *et al.*, 1999] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:1999–240, 1999.

[Donini *et al.*, 1991] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. Allen Sandewall, R. Fikes, and E., editors, *2nd International Conference on Knowledge Representation and Reasoning, KR-91*. Morgan Kaufmann, 1991.

[Fensel *et al.*, 2000] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. Oil in a nutshell. In *12th International Conference on Knowledge Engineering and Knowledge Management EKAW 2000*, Juanles-Pins, France, 2000.

[Heinsohn *et al.*, 1994] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2):367–397, 1994.

[Horrocks, 1995] Ian Horrocks. *A Comparison of Two Terminological Knowledge Representation Systems*. Master's thesis, University of Manchester, 1995.

[Karp, 1993] Peter D. Karp. The design space of frame knowledge representation systems. Technical Note 520, AI Center SRI International, May 5 1993.

[McGuinness *et al.*, 2000] D. McGuinness, R. Fikes, D. Connolly, and L.A. Stein. Daml-ont: An ontology language for the semantic web. *IEEE Intelligent Systems*, 2000. Submitted to Special Issue on Semantic Web Technologies.

[Staab and Maedche, 2000] Steffen Staab and Alexander Maedche. Ontology engineering beyond the modeling of concepts and relations. In *Proceedings of he ECAI'2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, Berlin, Germany, 2000.

A Modest Proposal: Reasoning Beyond the Limits of Ontologies

Wolfgang Wohner
Bavarian Research Center for Knowledge Based Systems
Orleansstraße 34, 81667 Munich, Germany
wohner@forwiss.de

Abstract

We will present an approach that extends the formal model of ontologies by application semantics. The novel notion of *laws* governing these semantics is motivated and introduced.

1 Introduction

In this short paper we want to stress the need for a framework that models the inference semantics of ontologies. An ontology provides a formalization of the concepts of an application area and their semantics, indicated e.g. by relations or axioms, but it is lacking a description of how this knowledge may be used for automated reasoning. We suggest to regard ontologies as static formal models that require additional information, i.e. *metadata on ontologies*, in order to be processed correctly. For explanatory purposes we will consider an exemplary ontology used for intelligent searching in semi-structured documents. An extended example in section 2 will motivate considerations about the semantics the ontology has to cover. Section 3 discusses how this knowledge may be applied when processing user queries and argues that an explicit modeling of the underlying patterns and rules is necessary.

2 Seeking Wisdom

Suppose a computer scientist expert is looking for some specific information, say, about the nature of knowledge. As this is a very complex question she might want to consult a local philosopher. The only philosopher living nearby she has heard of is a Mr. Smith but, unfortunately, he is not listed in the phone book. Now, she is looking for his address and so it is only reasonable that she will try to find Web documents containing this information.

Most probably she will first use one or more *keyword-based search engines* such as Google or AltaVista. The computer scientist's task consists of finding adequate keywords to formulate her query. Although her actual interest lies in getting in touch with a (any) philosopher living *nearby* she cannot express this fact using keywords. Generally, drawbacks of the keyword-based approach concern (i) the limited expressiveness of the query languages and (ii) the insufficient treatment of semantic text properties such as linguistic diversity or contextual semantics.

The computer science expert might therefore turn to *information retrieval (IR)* techniques like text mining and information extraction using wrappers. Although *text mining* techniques have been proven to yield acceptable results in certain application areas they are still very limited as they are predominantly concerned with exploiting linguistic features and not with the actual semantics of the text itself. Existing semantic analysis methods are less advanced and computationally too expensive to be used for exhaustive searching in large text corpora [Tan, 1999]. *Wrappers* on the other hand are used for selectively extracting textual components. But wrappers are highly specialized and will return useless results from pages (even valid ones) not complying to their templates, they focus on syntactic structure, not content and, consequently, wrappers know no mechanisms for adapting to different document structures as it is the patterns of these very structures (and not the associated concepts) they are looking for.

In summary, all approaches mentioned so far are lacking:

- a semantic notion of the components of a query (e.g. that 'Smith' is a name)
- a semantic notion of what the query expects as a return value (e.g. an address)
- a technique for adequately processing queries (e.g. adaptively, by semantic query rewriting)
- a general means for extracting the required information from heterogeneous text sources

Common to all of these requirements is the basic need for a sound and explicit modeling of background knowledge. A promising approach can be found in the context of database system design. The information stored in a database is highly structured according to its *schema*, an elaborate abstraction of some application area that has been formalized using e.g. entity/relationship (E/R) techniques. Each data unit of a database is strictly typed, e.g. (using relational syntax) the name 'Smith' might be a string value

of an attribute *surname* that appears in a relation called *philosophers*. An according database schema then allows for queries like (supposing the *philosophers* relation also contains an attribute *address*):

```
SELECT   address
FROM     philosophers
WHERE    surname = 'Smith'
```

Thus the internal structure of a database as depicted by its schema offers powerful querying possibilities: concepts like *surname* can be addressed directly and their semantics are known from the database system design. Nevertheless, there remains a remarkable gap between the homogeneous and well-structured data inside a database system and the heterogeneous, at best semi-structured sources of information found elsewhere, which renders integrating their semantics a complicated and complex task.

Heterogeneity, here, refers to differences in both, internal structure and vocabulary of the documents containing information. Ultimately, the gap between syntax and semantics has to be bridged. This can be facilitated significantly by taking advantage of the properties of markup languages (HTML, XML, SGML) that are used to describe metadata which is structuring and commenting on the textual content of documents. Metadata by itself cannot be directly identified with semantics (after all metadata is still data) but (i) it conforms to a predefined vocabulary and (ii) exhibits structural properties (e.g. nested structures) and these characteristics can be exploited to derive semantics.

The foundations for processing factual knowledge are addressed in the field of ontology engineering. *Ontologies* comprise an abstract knowledge representation of a certain domain. Modeling primitives are concepts, relations, functions, axioms and instances [Gruber, 1993] which are used to formalize the static aspects of the respective domain. There are two general approaches to combine ontologies and markup languages: (i) defining new markup which is directly related to the ontology or (ii) translating foreign markup into native concepts of the local ontology. The first approach has been propagated by SHOE [Luke and Heflin, 2000] and Ontobroker [Fensel *et al*., 1998] but its drawback is obvious. Since their markup methods did not evolve to become widely accepted standards, only a small portion of Web documents comply with them. For this reason current research, e.g. [Fensel *et al.*, 2000], [Farquhar, 1996], [Stuckenschmidt and Wache, 2000], is focused on establishing a direct linking between domain knowledge and various ways to express it because this provides the basis for reasoning on information which is distributed over a heterogeneous environment such as the semi-structured document space of the Web. The remainder of this paper will motivate a framework that is aimed at providing a formal basis for such reasoning processes which, eventually, could help the computer science expert find the philosopher's address.

3 Paving the Path

In this section we will examine dynamic aspects of ontology processing. An exemplary system used for providing access to heterogeneous semi-structured data sources will illustrate our approach. Basic assumptions about the system are:

- The system possesses a global ontology that comprises formalized knowledge about a domain.
- There is a set of heterogeneous semi-structured documents (e.g. XML documents) covering topics of that domain.
- There exists a mapping between markup tags of the documents and the concepts of the ontology, i.e. the ontology can 'understand' markup semantics in a sense that the concepts involved are part of its formal model.

The system's main purpose is to answer user queries about the contents of the documents. Return values can be document fractions (e.g. concepts, their values or combinations thereof) or complete documents. In order to retrieve valid results the system first has to understand the semantics of the query and then make use of the ontology's domain knowledge for exploring the syntactic structures of the documents. The general task is to derive information (semantics) from semi-structured data (data conforming to syntax). There are some properties of semi-structured data the system may take advantage of. We will illustrate this by referring to XML syntax:

- *Syntax definition*: the syntax definition of markup elements used within an XML document is known via its DTD, so the system is aware of all element names, their attributes and subelements.
- *Concepts*: the semantics of the structuring elements (tags) are known to the system because of the mapping between elements and ontology concepts.
- *Context*: markup elements are organized hierarchically thus establishing contexts (e.g. by nesting tags like <Name> and <Address> into <Person>) which can be interpreted semantically.
- *Types*: in a weak sense each markup element represents a type of its own but it is also possible to introduce primitive or derived element datatypes using e.g. XML Schema.

This syntax information can be utilized when processing queries that work on semi-structured documents. Existing systems, like On2broker [Fensel *et al.*, 2000], that provide access to semi-structured information sources are dealing

with this task but the inference mechanisms and heuristics applied here are usually hidden within their software components. We want to stress the importance of uncovering the underlying semantics and integrating them into the ontology structure. This is not just a matter of rendering implicit processes explicit but of providing a formal semantic model *about the usage* of the semantics an ontology provides on its part. Thus, such a formalization defines *metadata* about the ontology, foremost semantic processing rules we call *laws*. Again, laws have to be understood and executed by software components but the invaluable benefit they could provide is a homogeneous formal description of the semantic and syntactic implications of such processes.

Laws may be regarded as function templates that accept *cases* (e.g. a query) and contain formalized descriptions how to solve them. Our framework is aimed at defining a theoretical basis for such ontology laws and their impact on other elements of the ontology. For the remainder of this section we will stress various aspects of laws by referring to the illustrative example of the previous section.

- Laws address inference semantics.

The original query, '*Find the address of a philosopher living nearby'*, contains an inexact, or *vague*, concept: *nearby*. The meaning of *nearby* depends on the context of the query, as there are different notions of closeness in the context of houses and, say, atoms. In such cases techniques are needed to establish context which requires laws that describe how the desired information can be deduced. These techniques may vary for different semantic classes, or *categories*, of concepts, such as precise and vague ones, i.e.

- Laws can be general or attributed to single concepts or concept categories.

It is of major importance to identify such categories in order to establish a formal basis for reasoning processes. Once the category of a concept is known all laws attributed to that category can be directly applied to this concept as well.

- Laws state the limits of ontologies.

Some knowledge cannot be deduced because of incomplete knowledge. Although the context of *nearby* may be correctly inferred the point of reference (e.g. the computer scientist's own address) remains unknown. This indicates incomplete knowledge about the defining constituents of the query, i.e. at least one input factor of the respective law is missing and there is no other law describing how to compute it. Similarly, the ontology itself might be lacking concepts as well, e.g. a notion for closeness within the context of addresses might not be included. Generally, laws address *representational limits*, i.e. what can be expressed by an ontology, and *inferential limits* about what can be deduced from these representations.

- Laws control semantic query rewriting.

Automated *semantic query rewriting* is a promising technique for improving query return values. Using ontology knowledge an original query may be transformed into a set of refined queries. The excerpt of an XML document shown below does not contain an <Address> tag, so a query restricted to searching addresses would omit this document:

```
<Person>
   <Name> Smith </Name>
   <Phone> (222) 333-4444 </Phone>
   <Profession> philosopher </Profession>
</Person>
```

By contrast, laws provide rules for extending the scope of the query from addresses to e.g. phone numbers, street names and other address components known to the ontology. This would yield Mr. Smith's phone number, valuable information that the original query could not have produced.

- Laws manage uncertainty.

Uncertainty may play an important role in the context of iterative document querying, i.e. reasoning on grounds of intermediate results extracted from texts. From the XML example shown above it can be inferred that 'philosopher' is an instance of the concept *profession*. The value 'philosopher' can now be interpreted as a concept as well. But as this information has been derived from the textual content of a document it must be regarded as uncertain knowledge. Markup elements, on the other hand, can be mapped to concepts directly and therefore establish reliable knowledge. Uncertain knowledge is an omnipresent factor in intelligent information management and we will intensify our research efforts in that direction.

4 Conclusions and Future Work

We have motivated the importance of a framework for classifying and representing ontology laws and discussed some possible applications. Our future work will consist of elaborating this approach by providing a sound formal foundation of such a framework and incorporating a basic set of laws into the ontology of the intelligent information management system we are currently developing.

5 References

[Farquhar, 1996] A. Farquhar, R. Fikes, J. Rice. The Ontolingua Server: A tool for Collaborative Ontology Construction. *Proceedings of KAW96*. Banff, Canada, 1996.

[Fensel *et al.*, 1998] D. Fensel, S. Decker, M. Erdmann, R. Studer. Ontobroker: The Very High Idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, Sanibel Island, Florida, USA, pp. 131-135, May 1998.

[Fensel *et al.*, 2000] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, A. Witt. On2broker: Semantic Access to Information Sources at the WWW. In *Proceedings of IJCAI-99 Workshop on Intelligent Information Integration*, Stockholm, 31 July 1999.

[Gruber, 1993] R. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition #5*, pp. 199-200, 1993.

[Luke and Heflin, 2000] Luke, S., Heflin J. SHOE 1.01. Proposed Specification. SHOE Project. February 2000. http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm

[Stuckenschmidt and Wache, 2000] Context Modeling and Transformation for Semantic Interoperability. In *Proceedings of the 7th International Workshop on Knowledge Representation meets Databases (KRDB 2000)*, Berlin, Germany, August 21, 2000.

[Tan, 1999] A.-H. Tan. Text Mining: The state of the art and the challenges. In *Proceedings of the PAKDD'99 workshop on Knowledge Discovery from Advanced Databases*, Beijing, pp. 65-70, April 1999.

Joint Session with IJCAI-01 Workshop on e-Business and the Intelligent Web

Solving integration problems of e-commerce standards and initiatives through ontological mappings

Oscar Corcho, Asunción Gómez-Pérez
Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo s/n. Boadilla del Monte, 28660. Madrid. Spain.
{ocorcho, asun}@fi.upm.es

Abstract

The proliferation of different standards and joint initiatives for the classification of products and services (UNSPSC, e-cl@ss, RosettaNet, NAICS, SCTG, etc.) reveals that B2B markets have not reached a consensus on coding systems, level of detail of their descriptions, granularity, etc. This paper shows how these standards and initiatives, which are built to cover different needs and functionalities, can be integrated using a common multi-layered knowledge architecture through ontological mappings. This multi-layered ontology will provide a shared understanding of the domain for applications of e-commerce, allowing information sharing and interoperation between heterogeneous systems. We will present a tool called *WebPicker* and a method for integrating these standards and initiatives, enriching them and obtaining the results in different formats using the *WebODE* platform. As an illustration, we show a case study on the computer domain, presenting the ontological mappings between UNSPSC, e-cl@ss, RosettaNet and an electronic catalogue from an e-commerce platform.

1 Introduction

The popularity of Internet and the huge growth of new Internet technologies have led in the last years to the creation of a great amount of e-commerce applications ([McGuinness, 99] [Fensel, 00] [Berners-Lee, 99]). However, technology is not the unique key factor for the development of current e-applications. The context of e-commerce, and especially the context of B2B (Business to Business) applications, requires that an effective communication between machines is possible. In other words, semantic interoperability between the information systems involved in the communication is crucial.

Two extremely important factors contribute to this effective non-human communication: (1) a common language in which the resources implied in the communication can be specified, and (2) a shared knowledge model and vocabulary between the different systems that are present in the whole process. We will call them the *syntactic* and *semantic* dimensions.

The first dimension has led to the creation of varied representation languages for the specification of web resources (XOL, SHOE, OML, RDF, RDF Schema, OIL and DAML+OIL). A comparative study of the expressiveness and reasoning mechanisms of these languages can be found in [Corcho et al, 00].

The semantic dimension is related with the knowledge model and vocabulary used by the systems involved in the communication. In that sense, the use of a shared and common knowledge model and vocabulary increases the interoperability among existing and future information systems. This problem can be solved by ontologies. In fact, ontologies can be defined as "formal[1] and explicit specifications of a shared conceptualization" [Studer et al, 98]. If we compare this definition with the one given for the Semantic Web in [Berners-Lee, 99] ("the conceptual structuring of the Web in an explicit machine-readable way"), we can foresee that ontologies will play a key role in its development, and hence they will be applied to the key areas of the Semantic Web: e-commerce among others.

Large and consensuated knowledge models for e-commerce applications are difficult and expensive to build. Several standards and initiatives (UNSPSC, RosettaNet, e-cl@ss, NAICS, SCTG, etc[2]) came up in the previous years to ease the information exchange between customers and suppliers, and between different suppliers, by providing frameworks to identify products and services in global markets. However, the proliferation of standards and initiatives reveals that B2B markets have not reached a consensus on coding systems, level of detail, granularity, etc., which is an obstacle for the interoperability of applications following different standards. For instance, an application that uses the UNSPSC code cannot interoperate with an application that follows the e-cl@ss coding system. Consequently, we claim that with the current state of affairs it is more suitable to establish ontological mappings between existing standards and initiatives than to pretend to build *the* unified knowledge model from scratch.

[1] Formal must be understood as machine-readable.

[2] UNSPSC(http://www.unspsc.org), e-cl@ss(http://www.eclass.de) RosettaNet (http://www.rosettanet.org/), NAICS (http://www.census.gov/epcd/www/naics.html), SCTG (http://www.bts.gov/programs/cfs/sctg/welcome.htm).

Several architectures for the Semantic Web have arisen recently. Examples can be found in [Ambroszkiewicz, 00], for solving semantic interoperability to assure a meaningful interaction between heterogeneous agents, [Melnik et al, 00], where a layered architecture is proposed to solve the interoperability of different Web information models and [Benslimane et al, 00], where a multi-layered ontology definition framework is presented in a urban management application.

1.1 Aim of this paper

In this paper, we will focus on the semi-automatic integration of existing standards and initiatives in a multi-layered knowledge model for e-commerce applications through ontological mappings. We import semi-automatically standards and joint initiatives into the *WebODE* platform [Arpírez et al, 01] using the tool *WebPicker*, we integrate[3] them by means of ontological mappings, and enrich the unified knowledge model using *WebODE*. The resulting multi-layered knowledge architecture can be exported partially or completely into different representation languages (XML, RDF(S) and OIL).

The final multi-layered knowledge model will allow the intra-operability of vertical markets in specialized domains and also the inter-operability between different vertical markets (also known as horizontal markets).

The logical organization of the contents of the paper is as follows: Section 2 outlines the main steps of the proposed method, providing a global view of the whole process. Section 3 describes the standards and initiatives that we have selected as sources of information, as well as a product catalogue from an e-commerce platform. Section 4 describes the *WebODE* platform, which gives support for our method. In section 5, we describe briefly the tool *WebPicker* and the process of semi-automatic extraction of knowledge from the different sources of information. Section 6 deals with the final knowledge architecture that integrates the different proposals, paying special attention to the mappings between different layers of ontologies. Section 7 presents the main guidelines we have followed for ontology integration and enrichment. Section 8 deals with the automatic implementation in different languages from partial or global views of the ontologies. Finally, sections 9 and 10 will present the main conclusions that can be extracted from the work performed and future lines of work.

2 A method for reusing standards and initiatives to create e-commerce ontologies

In this section, we will explain the main steps of the method we propose for building e-commerce ontologies from standards and initiatives:

1. **Selection of standards, joint initiatives, laws, etc., of classification of products and services**. In this step, we select the sources of information that we consider relevant for our domain, from existing global or more specific agreements on classifications of products and services. They usually provide a commonly agreed taxonomy of products and/or services, which usually offers from 2 to 5 levels of depth.
2. **Knowledge models extraction**. This step semi-automates the process of knowledge acquisition from the sources of information previously selected and adapts them to the *WebODE's* knowledge model, which can be expressed in XML. This activity is performed using the tool *WebPicker*. Finally, the import service of *WebODE* is used to upload them into the platform.
3. **Design of a multi-layered knowledge architecture**. Taking into account features of the selected sources of information (covering, globality, specificity, etc), the aim of this step is the identification of relationships between components in the different taxonomies.
4. **Integration of knowledge models**. Knowledge models that have been automatically imported into the *WebODE* platform are integrated in the layered architecture, using the ontological mappings identified at the design phase.
5. **Enrichment of the integrated ontology**. Current standards do not include attributes for products, relations between products, disjoints nor exhaustive knowledge, functions, axioms, etc. Most of them just represent taxonomies of concepts, and other ones just include some attributes for them. Hence, they can be enriched with extra information when possible.
6. **Ontology exportation**. The whole ontology or specific parts of the ontology can be exported into different kinds of languages, so that they can be tractable by the systems that are using it for any application.

The following sections will describe this method and will apply it to a case study in the computers domain.

3 E-commerce standards as knowledge models

Standards, joint initiatives, laws, etc., are good sources for ontology building, since they are pieces of information that have been agreed by consensus or are followed by a community.

In this section, we present three proposals for the classifications of products that have arisen in the context of e-commerce: UNSPSC, RosettaNet and e-cl@ss. These initiatives are being developed to ease the information exchange between customers and suppliers, and between suppliers, by providing consistent, standardised frameworks to identify products and services in a global market.

Other similar approaches exist and are available (NAICS, for US, Canada and Mexico, SCTG for transporting goods, etc). We have just selected the ones enumerated before to show the adequacy of our work in this context.

[3] We talk about integration of ontologies instead of merge because we do not pretend to build a single knowledge model out from the existing ones, but preserve them in a common architecture.

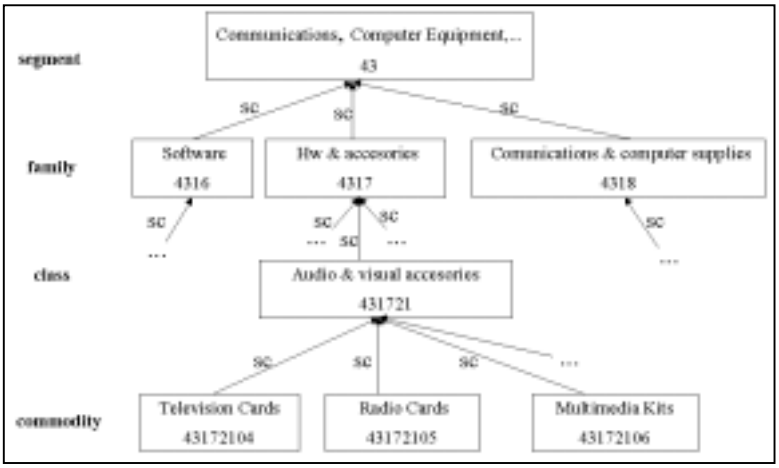


Figure 1. A snapshot of the classification of UNSPSC for computer equipment.

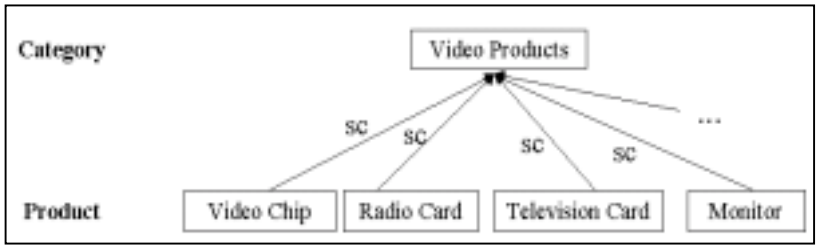


Figure 2. A snapshot of the classification of video products of the RosettaNet taxonomy.

Finally, we present an electronic catalogue from an e-commerce platform, which fits in the ontology architecture.

3.1 UNSPSC (Universal Standard Products and Services Classification Code)

UNSPSC is a non-profit organisation composed of partners such as 3M, AOL, Arthur Andersen, BT, Castrol and others.

Its coding system is organised as a five-level taxonomy of products, each level containing a two-character numerical value and a textual description. These levels are defined as follows:

- **Segment.** The logical aggregation of families for analytical purposes.
- **Family.** A commonly recognised group of inter-related commodity categories.
- **Class.** A group of commodities sharing a common use or function.
- **Commodity.** A group of substitutable products or services.
- **Business Function.** The function performed by an organisation in support of the commodity. This level is seldom used.

The current version of the UNSPSC classification contains around 12000 products organized in 54 segments. Segment 43, which deals with computer equipment, peripherals and components, contains around 300 kinds of products.

Figure 1 shows a small part of the UNSPSC classification, related to computer equipment (segment 43 of the UNSPSC classification).

The main drawbacks of UNSPSC are: (a) the lack of vertical cover of the products and services which appear in the classification; (b) the lack of attributes attached to the concepts that appear in the taxonomy[4]; (c) the design of the classification without taking into account the inheritance between the products that are described; (d) the non-providing different views of the classification, taking into account cultural and social differences, where classifications could be made in different ways than the ones presented in this standard.

3.2 RosettaNet Technical Dictionary

RosettaNet is a self-funded, non-profit consortium composed of several information technology and electronic components companies. Therefore, this classification is just focused on electronic equipment.

RosettaNet classification does not use a numbering system, as UNSPSC does, but it is just based on the names of the products it defines. This classification is related to the UNSPSC classification by providing the UNSPSC code for each product defined in it.

[4] Initiatives such as UCEC (*Universal Content Extended Classification*) are trying to solve this problem by adding attributes to the concepts in the last level of the taxonomy. However, they are not freely available.

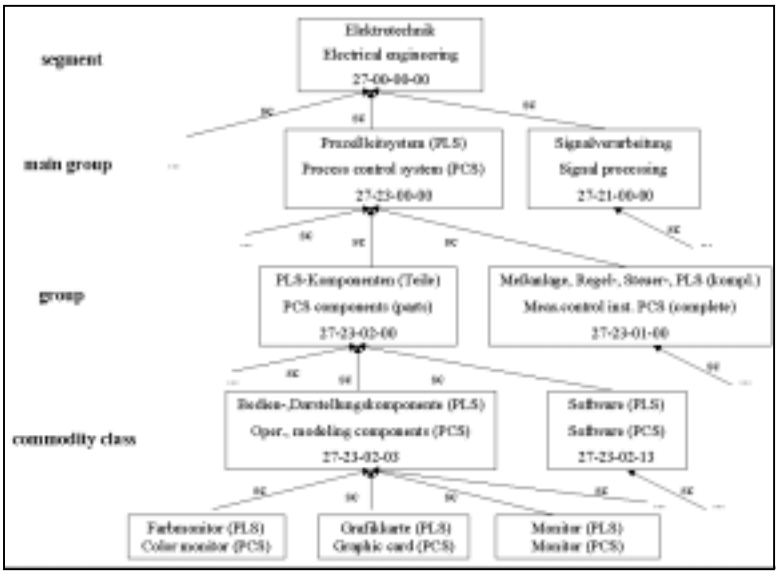


Figure 3. A snapshot of the classification of e-cl@ss for electrical engineering products (in German and English).

RosettaNet has just two levels in its taxonomy of concepts:

- **RN Category**. Group of products (i.e., *Video Products*)
- **RN Product**. Specific product (i.e., *Television Card*, *Radio Card*, etc.).

The RosettaNet Technical Dictionary classification consists of 14 categories and around 150 products. It must be taken into account (in relationship with UNSPSC) that RosettaNet just deals with the electronic equipment domain, which is more specific than the UNSPSC classification.

Figure 2 shows part of the RosettaNet classification, related to video products for computer equipment.

The main drawback of this taxonomy is that there are only two levels of classification, which implies that the structure of the taxonomy is very simple. This classification also shares some of the problems of UNSPSC, namely, lack of attributes and design without taking into account inheritance in the taxonomy of concepts.

The problem of using this classification in a vertical market is partially solved, as it is focused on the specific domain of electronic equipment, although it just offers a low level of detail in this domain.

3.3 E-cl@ss

E-cl@ss is a German initiative to create a standard classification of material and services for information exchange between suppliers and their customers. In fact, it is similar to the UNSPSC initiative, and will be used by companies like BASF, Bayer, Volkswagen-Audi, SAP, etc.

The e-cl@ss classification consists of four levels of concepts (called *material classes*), with a numbering code similar to the ones of UNSPSC (each level has two digits that distinguish it from the other concepts). These levels are: *Segment*, *Main group*, *Group* and *Commodity Class*.

e-cl@ss levels are equivalent to the first four ones provided in UNSPSC; hence, they are not described any further. Finally, inside the same commodity class we may find several products (in this sense, several products can share the same code, and this could lead to a fifth level with all of them, as it can be seen in figure 3).

It also contains around 12000 products organized in 21 segments. Segment 27, which deals with *Electrical Engineering*, contains around 2000 products. Finally, the main group 27-23, which deals with *Process Control Systems*, together with the main groups 24-01 to 24-04, which deal with *Hardware*, *Software*, *Memory* and other computer devices, contain around 400 concepts.

This classification suffers from the same drawbacks as UNSPSC. In fact, it is a similar approach, although within a smaller social environment, as it will be used by German companies. Additionally, terms and their descriptions are written both in English and German.

3.4 E-commerce platform catalogue

We have selected a catalogue of products from an existing e-commerce platform that deals with computer equipment, so that we have found a common domain to show a whole case study in this paper.

This catalogue is structured in two kinds of elements, called categories and items (very similar to the RosettaNet structure). Catalogue items are actual products sold by the e-commerce platform. Attributes are defined on them with the main characteristics of each product. Categories are groups of products (items) or groups of other categories. They are created with the aim of grouping products taking into account factors such as marketing, common uses, etc. They do not have attributes defined on them.

The selected catalogue contains around 400 items, with 2/3 levels of depth in the hierarchy of categories. Figure 4

shows some elements in the catalogue.

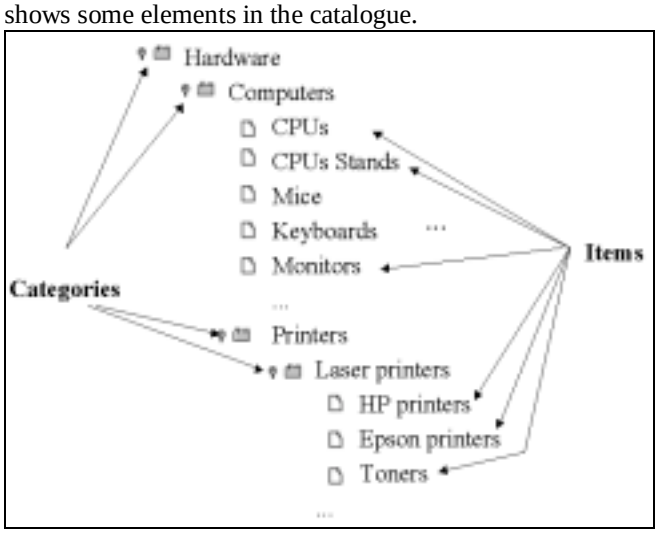


Figure 4. A snapshot of some elements in the catalogue.

In contrast with the classifications presented before, catalogues cannot be considered themselves as good sources of information for building ontologies, as they are not shared by a community nor represent any consensus. They are designed instead as classifications of products and services from the market point of view.

However, catalogues play an important role in the whole e-business process: they present the set of products offered by an e-commerce application and they are the front-end in the exchange of products in B2C and B2B environments.

4 WebODE

WebODE [Arpírez et al, 01] is an ontological engineering platform that allows the collaborative edition of ontologies at the conceptual level, providing means for their automatic exportation and importation in XML and their translation into and from varied ontology specification languages.

WebODE's conceptual model is based on the intermediate representations of METHONTOLOGY [Fernández et al, 99], allowing for the representation of concepts and their attributes (both class and instance attributes), taxonomies of concepts, disjoint and exhaustive knowledge, ad-hoc relations between concepts, constants, axioms and instances.

The conceptualization phase of ontologies is aided by both a HTML form-based and a graphical user interfaces, a user-defined-views manager, a consistency checker for the components defined in the ontology, an inference engine implemented in Prolog to perform inferences with the information provided, an axiom builder to assist the creation of these components and a the documentation service.

The platform is built upon an application server, which provides high extensibility by allowing the addition of new services and the common use of services provided by the platform. Examples of these services are the catalogue manager, the taxonomy merger and *WebPicker*, which is presented in the next section.

5 *WebPicker*: obtaining knowledge models from structured information

The classifications described in the previous section are represented using different representation formats. UNSPSC is available in HTML (taxonomies are presented visually); RosettaNet is in HTML, XML and Microsoft Excel, and e-cl@ss is available in Microsoft Excel; finally, the catalogue is available in XML.

If we want to work with all this information together, we should use a common representation format for it, so that the treatment of this information can be performed homogeneously, no matter what its origin is. We have decided to use the *WebODE* knowledge model [Arpírez et al, 01] as the reference model where all the information will be translated to.

In [Corcho et al, 01], we present in detail *WebPicker* and the different processes we have followed to translate the contents of the different sources of information into *X-WebODE*, the XML syntax of *WebODE*, so that we have been able to import them into the platform. As an illustration, we present figure 5, which shows a summary of the process of importing UNSPSC[5] into *WebODE*.

The figure shows that UNSPSC information is available in several HTML pages, one per UNSPSC segment. Once identified the valuable information in each page, it was extracted with *WebPicker*, which converted it into XML, and finally, all the XML documents were included in a single XML document that followed the grammar defined in the *WebODE* DTD [Arpírez et al, 01].

The classification was uploaded into the WebODE platform using its XML import facility.

The processes applied for RosettaNet, e-cl@ss and the catalogue were very similar.

6 Multi-layered ontology architecture design

Before describing our contribution to ontology architectures, we will revise briefly some important pieces of the state of the art in the classification of ontologies.

Till now, many different types of ontologies have been identified and classified. [Mizoguchi et al, 95] distinguish between domain ontologies, common-sense ontologies, meta-ontologies and task ontologies. [Van Heijst et al, 97] classify ontologies using two dimensions: the amount and type of structure and the subject of the conceptualization. Terminological, information and knowledge modeling ontologies usually have a richer internal structure, and they belong to the first dimension. In the second dimension, they distinguish application, domain, generic and representation ontologies. A common framework for understanding both classifications in a unified manner is shown in figure 6.

[5] UNSPSC transformation allowed us to detect missing pieces of information in the HTML pages and errors on the numbering of some products that were reported to the UNSPSC responsible.

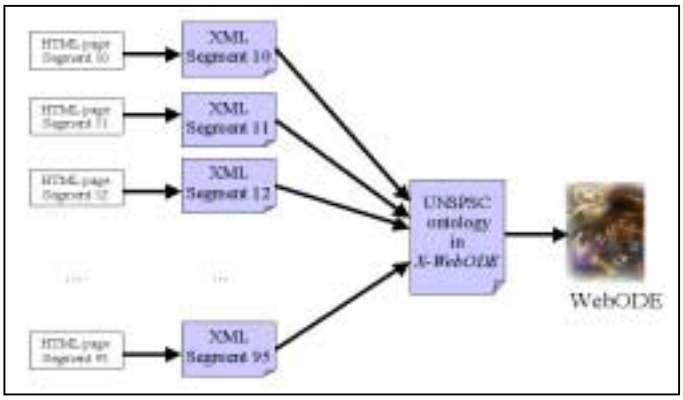


Figure 5. The process of importing UNSPSC into *WebODE*.

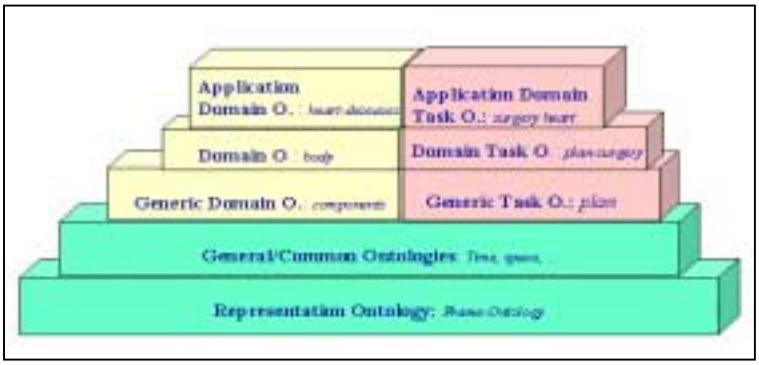


Figure 6. Libraries of ontologies.

Figure 6 also shows that ontologies are usually built on top of other ones (application domain ontologies on top of domain ontologies, domain ontologies on top of generic domain ontologies, and so on). This layered approach for the building of ontologies makes it easier their development, taking into account the following design criteria:

- Maximum monotonic extensibility [Swartout et al, 97] [Gruber, 93], as new general or specialized terms can be included in the ontology in such a way that it does not require the revision of existing definitions.
- Clarity [Gruber, 93], as the structure of terms implies the separation between non similar terms (common-sense terms vs. specialized domain ontologies).

6.1 A proposal for a multi-layered architecture of e-commerce ontologies

Our approach consists of structuring our ontologies in several layers, following the criteria presented above. This architecture will be illustrated with examples taken from the sources of information presented in section 3.

Figure 7 shows the ontological mappings that can be established between ontologies present in the architecture.

In this sense, we propose a common *upper level ontology*, which defines the common terms used in the communication between systems, providing a unified upper-level vocabulary for all the systems accessing the ontology.

Generic e-commerce ontologies provide broad, coarse-grained classifications of products and services in the e-commerce domain.

More specialized ontologies (*regional e-commerce ontologies*) can be created for the different domains that will be handled by the different systems (electronic equipment, tourism, vehicles, etc). The concepts of these ontologies will be mapped to the concepts in the generic e-commerce ontologies, so that they share a common root for all the concepts. These ontologies can be organized in as many layers as the ontology developers consider necessary.

Optionally, very specialized *local e-commerce ontologies* could be created for each one of the systems that access to the whole structure of the knowledge (electronic equipment companies, tourism companies, vehicle manufacturers, etc).

Finally, the lowest level (below *local e-commerce ontologies*) will contain the catalogues, with their products (*items*) and groups of products (*categories*) linked to one or more concepts at any level of the whole ontology (preferably the most specific ones).

As set out before, this layered approach will allow the intra-operability of vertical markets in specialized domains and also the inter-operability between different vertical markets (also known as horizontal markets).

6.2 A case study in the computers domain

Considering the main features of the standards and initiatives that we have selected for this study and imported

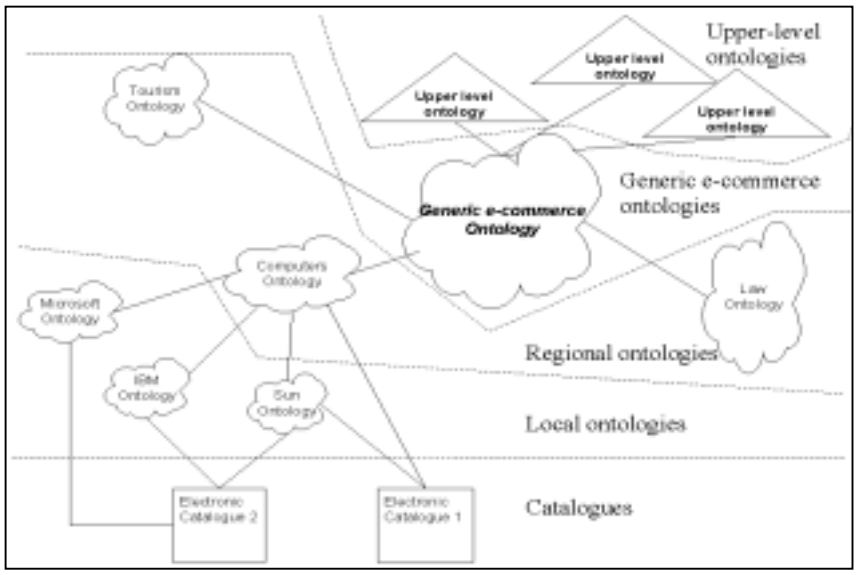


Figure 7. General ontological mappings between ontologies, and between ontologies and catalogues.

into *WebODE*, we can try to fit them in the proposed architecture, with the following roles for each of them[6]:

UNSPSC can act as a generic e-commerce ontology, where a coarse-grained classification of products and services is offered. Hence, it can provide the roots for all the products and services that will be inserted in the different regional and local ontologies that use it, and could be also interesting to use it for allowing the interoperability between different vertical markets (because of its wide covering of products and services).

The same applies to e-cl@ss, whose development is being performed following a similar set of criteria. In this sense, both classifications share most of the products and services, although they are classified in different ways.

Finally, RosettaNet will play the role of a *regional ontology* in the domain of electronic equipment, focusing on this particular business area, although not presenting too much detail on the components that can be sold/bought/exchanged.

More regional ontologies could be created below RosettaNet (for instance, regional ontologies for computer manufacturers, hi-fi equipment, electrical device manufacturers, etc.), and local ontologies could be also created: for instance, one local ontology for each specific company in each of the business sectors identified above (IBM, HP, Sun, etc.).

Finally, we have to take into consideration the role of the catalogue presented in section 3.4. Its items and categories are mapped to concepts in the ontology. Using these mappings, we will be able to access the attributes of any product through the taxonomy of concepts of the ontology, we will be able to perform reasoning with the information represented in the ontology, we will facilitate searches of products from many different points of view, etc.

Figure 8 summarizes the ontological mappings between the standards and between the standards and catalogues in the context of the architecture proposed in this paper.

Please note that we present two generic e-commerce ontologies in our example. This fact enforces the idea of facilitating searches of products using different points of view, as products will commonly be classified with respect to the different standards and initiatives, and ontological mappings between both of them will be also established. Communication between systems using the ontologies in this architecture is still good, though providing much richer information on products that are placed in its lowest levels.

An additional remark must be made on the flexibility of this architecture. In case we want to include another classification in it, we shall study its characteristics and decide the level it should be placed in. The structure we present in figure 8 is adapted for this case study, but new ontologies could appear above our current generic e-commerce ontologies and additional intermediate levels in the regional or local ontologies area could also appear.

7 Ontology integration and enrichment

7.1 Ontology integration

Once sketched the similarities and differences between the standards described and the role of each of them in the multi-layered architecture proposed, we will make a detailed analysis of the relationships that can be established between their terminology.

1. We will start with the ontological mappings **between ontologies**, be them placed at the same level in the architecture or at different levels:

Equivalence mappings. They occur when a concept in the ontology is equivalent (or the most similar) to other concept

[6] There are no strict rules for the decision of the role of each classification in the overall architecture. It usually depends on its degree of generality and granularity

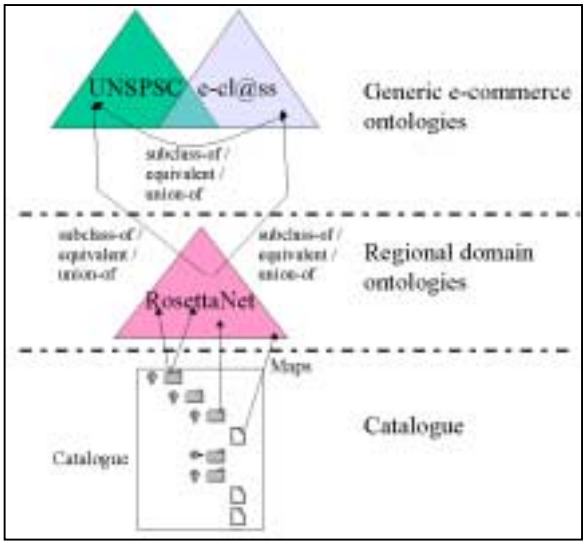


Figure 8. Ontological mappings between UNSPSC, e-cl@ss, RosettaNet and the catalogue.

or concepts in another ontology.

This ontological mapping is especially interesting between ontologies at the same level, as it allows interaction between systems using different standards or initiatives. It also provides several means of classifying products. For instance, concept *Diskette* in e-cl@ss (24-03-03-00) and *Floppy diskettes* in UNSPSC (43180601) are equivalent.

There are also equivalence mappings between concepts from ontologies in different layers, as it is shown in figure 9. For instance, concept *Monitor* in RosettaNet is equivalent to concept *Monitors* in UNSPSC (43172401).

As RosettaNet has already predefined the equivalence mappings between its concepts and concepts in UNSPSC, this task has been performed automatically with *WebPicker*. However, some of these equivalence mappings have been transformed into subclass-of ones after a detailed analysis of both standards, as it is shown in figure 9 with concepts *Video chip* in RosettaNet and *Hybrid Integrated Circuits* in UNSPSC (321017).

Subclass-of mappings. They occur when a concept in an ontology is a subclass of other concept or concepts in another ontology.

For instance, concept *Dot Matrix Printers* in UNSPSC (43172503) is subclass of concepts *Printer (PCS)* and *Printer (proc. comp.)* in the e-cl@ss classification (27-23-02-12 and 27-23-02-34).

This mapping can be also established between concepts in ontologies from different layers. For instance, concept *Laser Printer* in RosettaNet is also a subclass of *Printer (PCS)* and *Printer (proc comp)* in e-cl@ss classification (27-23-02-12 and 27-23-02-34).

An important remark must be made at this point. Brother concepts in an ontology do not have to share the same parent concepts in another ontology: classification criteria may be different in both ontologies.

Union-of mappings. They occur when a concept in an ontology is equivalent to the union of two or more concepts in another ontology.

For instance, concept *Monitors* in UNSPSC (code 42172401) is equivalent to the union-of concepts *Monitor (PCS)* and *Monitor* (codes 27-23-02-03 and 24-01-06-00, respectively) in e-cl@ss.

2. The second kind of ontological mappings that we have studied deal with **catalogues and ontologies**.

We have just considered *maps* between items (and categories) in the catalogue and concepts in the ontology: an item/category in the catalogue can be mapped to one or more concepts in the ontology (be it the local ontology, any of the regional ontologies or the generic e-commerce ontologies), stating that the item/category is defined by the concept(s) in the ontology to which it is linked.

The previous remark about subclass-of mappings between concepts in ontologies can also be applied to this case. Taking into consideration design issues of catalogues, it will be common to find items under the same category linked to very distant concepts in the ontology. For instance, let's suppose items in the catalogue that are grouped together because of their use: *laser printers* and *toners*. They will be probably mapped to very distant concepts in the ontology.

Other works on ontology integration have proposed their sets of inter-ontology relationships. For instance, the OBSERVER [Mena et al, 2000] system proposes synonym, hyponym, hypernym, overlap, disjoint and covering relationships between concepts in the same and different ontologies. DWQ [Calvanese et al, 98] proposes intermodel assertions such as subsetting, definition, completeness, synonym and homonym relationships.

Although terminology used in different projects is different, the meaning of these relationships is very similar to each other. In our work, we propose the equivalence relationship (which is named synonym in both projects), the

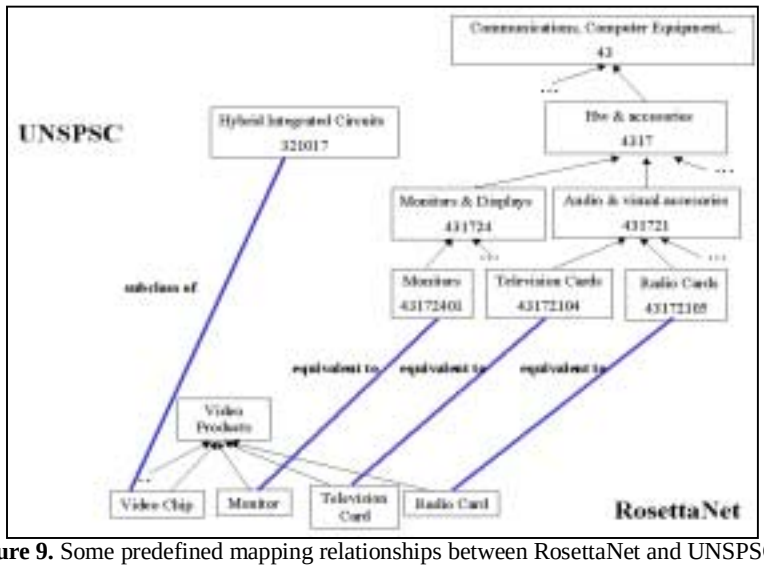


Figure 9. Some predefined mapping relationships between RosettaNet and UNSPSC.

subclass-of relationship (which is named hyponym and subsetting, respectively) and union-of (which is named covering and completeness). The rest of relationships are not important for our domain.

7.2 Ontology enrichment

Once all the classifications have been integrated in *WebODE*, the next phase consists of enriching them with new attributes for concepts, disjoints and exhaustiveness knowledge, relations, functions and axioms. This will make the resulting ontologies richer and will allow performing reasoning with the knowledge contained in them.

We are currently working on the enrichment of these classifications. First, we have focused on properties, taking into account several sources of information for creating them: properties for defining products that are provided by the RosettaNet IT and EC Technical Dictionaries; properties that we have found in several actual e-commerce catalogues from different companies and other common-sense properties that we consider interesting from both KR and marketing points of view. Unfortunately, we have not been able to use attributes from the UCEC classification for UNSPSC, because this information is not publicly available.

Work on taxonomies is also being performed. We are trying to identify and specify disjoint and exhaustive partitions between concepts, with the aim of making more robust taxonomies of concepts, as well as providing better search mechanisms for applications using these ontologies.

We will also focus on the most useful relations between concepts for e-commerce purposes, such as "concept X uses concept Y", "concept X and concept Y are used together", "concept X and concept Y have the same functionality", etc., as well as functions or axioms.

8 Ontology exportation

The last step of the method proposed in section 2 deals with the exportation of global or partial views of the ontologies to implementation code. This step is important, as it will generate the ontology in a format/code that is tractable for the systems involved in the application that justifies its use.

This exportation step is automatically performed using the translators provided by the *WebODE* platform (currently, XML, RDF(S) and OIL). These translators transform the ontologies conceptualized using the knowledge model of *WebODE* into the knowledge model of the target implementation language.

We may also choose whether exporting all the components in the ontologies or exporting just restricted sets of components, which the user can specify explicitly.

9 Conclusions

E-business applications are adopting standards and initiatives for allowing interoperation and interchange of information between information systems. Ontologies aim to provide a shared machine-readable view of domain knowledge, allowing information sharing for heterogeneous systems. In this paper, we have put together both areas, proposing a method for reusing and improving existing standards and initiatives for classification of products and services in the e-business domain creating of a multi-layered ontology that integrates them into a single architecture.

This paper shows how these standards and joint initiatives can be processed, transformed into knowledge models, integrated in a multi-layered architecture, enriched with new information and transformed again into implementation code suitable for its use by different systems.

From the e-business point of view, this approach offers the following advantages:

- Existing standards and initiatives are enriched with additional information that can be used for offering better services in e-business applications: deducting new information about products and customers, allowing a better search for products and services, etc.

- Multiple criteria for classificating a product or service.
- E-commerce catalogues can be integrated in the whole knowledge architecture, allowing a clear distinction between KR and marketing decisions.
- E-commerce catalogues are not necessarily built from scratch, as they can be built from the existing ontology and adapted later because of marketing decisions.

From the ontological engineering point of view, this approach offers the following advantages:

- Ontologies are not built from scratch. Their skeleton is built extracting relevant information from distributed sources that contain consensus knowledge. Hence, there is a great time reduction for knowledge acquisition and reaching consensus, ameliorating the KA bottleneck.
- Multiple views are allowed for any component in the ontology, in the sense that different generic ontologies can be selected, which will offer different sets of criteria for the classification of products and services.
- A knowledge architecture suitable for representing ontologies shared by e-commerce applications. It is based on a layered approach, which distinguishes global/widely-shared concepts, more domain specific ones and a final place for e-commerce catalogues.

From a technological point of view, we present *WebODE* as an ontological engineering platform that allows:

- Processing HTML pages, Excel documents, etc., and transform them into the *WebODE* knowledge model, using its specialized service *WebPicker*.
- Creating a multi-layered ontology through ontological mappings.
- Enriching ontologies with attributes, disjoints and exhaustive knowledge, relations, axioms, etc.
- Exporting the whole ontology or user-defined views into implementation code, suitable for other systems.

10 Future work

UPM participates in the EU-project MKBEEM (IST-1999-10589), which is building a mediation system for enabling online access to products and services in the customer's native language [Leger et al, 00]. The multi-layered knowledge architecture presented in this paper is used in this project for the representation of products and services offered in the catalogues of a B2B company.

Experience obtained in this project helped us identify the ontological mappings presented in section 7, and will help us identify more useful mappings between components in the same and different layers of the architecture. The use of this architecture will also aid the definition of many services that ontology servers must provide for applications in the Semantic Web (especially in the e-commerce domain).

Acknowledgements

This work is supported by a FPI grant funded by UPM and partially supported by the project *"ContentWeb"*, funded by *Ministerio de Educación y Ciencia*. We also thank Alberto Cabezas for implementing *WebPicker*, and Julio C. Arpírez, for the design and implementation of the *WebODE* platform.

References

[Ambroszkiewicz, 00] Ambroszkiewicz, S. *Semantic Interoperability in Agentspace*. Workshop on Semantic Web: Models, Architecture and Management. Lisbon. Sept. 2000.

[Arpírez et al, 01] Arpírez, J. *WebODE User Manual*. Technical Report. February, 2001.

[Benslimane et al, 00] Benslimane, D., Leclercq, E., Savonnet, M., Terrasse, M. N., Yétongnon, K. *On the definition of generic multi-layered ontologies for urban applications*. Computers, Environment and Urban Systems. #24. pp: 191-214. 2000.

[Berners-Lee, 99] Berners-Lee, T., Fischetti, M. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper. San Francisco. 1999.

[Calvanese et al, 98] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R. *Description Logic Framework for Information Integration*. 6th Intl. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98). 1998.

[Corcho et al, 00] Corcho, O., Gómez-Pérez, A. *A RoadMap to Ontology Specification Languages*. EKAW'00. October, 2000.

[Corcho et al, 01] Corcho, O., Gómez-Pérez, A. *Ontology acquisition and Integration from Web Environments using WebPicker*. 6th Intl. Workshop on Applications of Natural Language for Information Systems. Madrid. June, 2001.

[Fensel, 00] Fensel, D. *Ontologies: silver bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag. 2000.

[Fernández et al, 99] Fernández, M.; Gómez-Pérez, A.; Pazos, J.; Pazos, A. *Building a Chemical Ontology using Methontology and the Ontology Design Environment*. IEEE Intelligent Systems and their applications. #4 (1):37-45. 1999.

[Gruber, 93] Gruber, R. *A translation approach to portable ontology specification*. Knowledge Acquisition. #5: 1993.

[Leger et al, 00] Leger, A. and others. *Ontology domain modeling support for multi-lingual services in E-Commerce: MKBEEM*. ECAI'00 Workshop on Applications of Ontologies and PSMs. Berlin. Germany. August, 2000.

[McGuinness, 99]] McGuinness, D. *Ontologies for Electronic Commerce*. AAAI '99 Artificial Intelligence for Electronic Commerce Workshop, Orlando, Florida, July, 1999.

[Melnik et al, 00] Melnik, S., Decker, S. *A Layered Approach to Information Modeling and Interoperability on the Web*. Workshop on Semantic Web: Models, Architecture and Management. Lisbon. September, 2000.

[Mena et al, 00] Mena, E., Illarramendi, A., Kashyap, V., Sheth, A. *OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies*. International Journal Distributed and Parallel Databases (DAPD), 8(2), pp. 223-271, April 2000.

[Mizoguchi et al, 95] Mizoguchi, R.; Vanwelkenhuysen, J.; Ikeda, M. *Task Ontology for reuse of problem solving knowledge*. In N.J.I. Mars "Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing.". IOS Press. 1995.

[Studer et al, 98] Studer, R., Benjamins, R., Fensel, D. *Knowledge Engineering: Principles and Methods*. DKE 25(1-2).. 1998

[Swartout et al, 97] Swartout, B., Patil, R., Knight, K., Russ, T. *Toward Distributed Use of Large-Scale Ontologies*. Spring Symposium Series. 1997.

[Van Heijst et al, 97] Van Heist G., Schreiber A. Th., Wielinga B. J., *Using explicit ontologies in KBS development*. International Journal of Human-Computer Studies, 45, pp. 183-292, 1997.

Issues in Ontology-based Information Integration

Zhan Cui, Dean Jones and Paul O'Brien
Intelligent Business Systems Research Group
Intelligent Systems Lab
BTexact Technology

Abstract

Solving queries to support e-commerce transactions can involve retrieving and integrating information from multiple information resources. Often, users don't care which resources are used to answer their query. In such situations, the ideal solution would be to hide from the user the details of the resources involved in solving a particular query. An example would be providing seamless access to a set of heterogeneous electronic product catalogues. There are many problems that must be addressed before such a solution can be provided. In this paper, we discuss a number of these problems, indicate how we have addressed these and go on to describe the proof-of-concept demonstration system we have developed.

1. Introduction

There are a number of obstacles to completely open e-commerce over the Internet. One of the major problems is the vast amount of information that is available and our ability to make sense of it. For example, how do we identify whom to do business with? How do we know that a supplier's products are what we are looking for? It is only once we know what people are saying that we can start to identify who is worth talking to. In this article, we will discuss a number of related issues and describe the way we have begun to address some of them.

The problems of interoperability between interacting computer systems have been well documented. A good classification of the different kinds of interoperability problems can be found in [Sheth, 98] who identifies the system, syntactic, structural and semantic levels of heterogeneity. The system level includes incompatible hardware and operating systems; the syntactic level refers to different languages and data representations; the structural level includes different data models and the semantic level refers to the meaning of terms using in the interchange. A good example of semantic heterogeneity is the use of synonyms, where different terms are used to refer to the same concept. There are many more types of semantic heterogeneity and they have been classified in [Visser *et al.*, 1998]

Many technologies have been developed to tackle these types of heterogeneity. The first three categories have been addressed using technologies such as CORBA, DCOM and various middleware products. Recently XML has gained acceptance as a way of providing a common syntax for exchanging heterogeneous information. A number of schema-level specifications (usually as a Document Type Definition or an XML Schema) have recently been proposed as standards for use in e-commerce, including ebXML, BizTalk and RosettaNet. Although such schema-level specifications can successfully be used to specify an agreed set of labels with which to exchange product information, it is wrong to assume that these solutions also solve the problems of semantic heterogeneity. Firstly, there are many such schema-level specifications and it cannot be assumed that they will all be based on consistent use of terminology. Secondly, it does not ensure consistent use of terminology in the data contained in different files that use the same set of labels. The problem of semantic heterogeneity will still exist in a world where all data is exchanged using XML structured according to standard schema-level specifications.

A solution to the problems of semantic heterogeneity should equip heterogeneous and autonomous software systems with the ability to share and exchange information in a semantically consistent way. This can of course be achieved in many ways, each of which might be the most appropriate given some set of circumstances. One solution is for developers to write code which translates between the terminologies of pairs of systems. Where the requirement is for a small number of systems to interoperate, this may be a useful solution. However, this solution does not scale as the development costs increase as more systems are added and the degree of semantic heterogeneity increases.

Our solution to the problem of semantic heterogeneity is to formally specify the meaning of the terminology of each system and to define a translation between each system terminologies and an intermediate terminology. We specify the system and intermediate terminologies using *formal*

ontologies and we specify the translation between them using *ontology mappings*. A formal ontology consists of definitions of terms. It usually includes concepts with associated attributes, relationships and constraints defined between the concepts and entities that are instances of concepts.

We provide software support for the definition and validation of formal ontologies and ontology mappings, allowing us to resolve semantic mismatches between terminologies according to the current context (*e.g.* such as the application.) In the next section we discuss a number of issues relating to the use of ontologies in enabling semantic interoperability. We then describe how we have addressed some of these issues using a system called DOME (Domain Ontology Management Environment) which includes a set of tools for creating and mapping between ontologies, for browsing and customising ontologies and for constructing concept-based queries.

2. Issues in Resolving Semantic Heterogeneity

In this section we describe some of the problems involved in achieving semantic interoperability between heterogeneous systems.

2.1 Developing ontologies

In any reasonably realistic e-commerce scenario involving interoperability between systems, semantic heterogeneity is a significant problem and will continue to be so in the future. A solution to this problem based on the use of formal ontologies will need to accommodate different types of ontologies for different purposes. For example, we may have *resource ontologies*, which define the terminology used by specific information resources. We may also have *personal ontologies*, which define the terminology of a user or some group of users. Another type is *shared ontologies*, which are used as the common terminology between a number of different systems.

The problem of developing ontologies has been well-studied and a number of methodologies have been proposed. A comparative analysis of these can be found in [Jones *et al.*, 1998].) One of the major conclusions of this study was that the best approach to take in developing an ontology is usually determined by the eventual purpose of the ontology. For example, if we wish to specify a resource ontology, it is probably best to adopt a bottom-up approach, defining the actual terms used by the resource and then generalising from these. However, in developing a shared ontology it will be extremely difficult to adopt a bottom-up approach starting with each system, especially where there are a large number of such systems. Here, it is most effective to adopt a top-down approach, defining the most general concepts in the domain first.

2.2 Mapping Between Ontologies

In order to resolve the problems of semantic mismatches discussed above, we will often need to translate between different terminologies. While it would be ideal to be able to automatically infer the mappings required to perform such translations, this is not always possible. While the formal definitions in an ontology are the best specification of the meaning of terms that we currently have available, they cannot capture the full meaning. Therefore, there must be some human intervention in the process of identifying correspondences between different ontologies. Although machines are unlikely to derive mappings, it is possible for them to make useful suggestions for possible correspondences and to validate human-specified correspondences.

Creating mappings is a major engineering work where re-use is desirable. Declaratively-specifying mappings allows the ontology engineer to modify and reuse mappings. Such mappings require a mediator system that is capable of interpreting them in order to translate between different ontologies. It would also be useful to include a library of mappings and conversion functions as there are many standard transformations which could be reused *e.g.* converting kilos to pounds, etc.

Mapping between ontologies is not an exact science. Certain semantic mismatches cannot be resolved exactly but may involve some loss of information *e.g.* when translating from a colour system based on RGB values to one which uses terms such as ‘red’, ‘blue’, etc. Whether or not the loss of information is an issue varies between applications. In some domains, precision of information is more important than in others. For example, in e-commerce, imperfect information is generally unacceptable, whereas it is widely accepted that internet search engines will return many irrelevant results.

2.3 Ontologies and Resource Information

It is generally acknowledged that we have more information than we know what to do with. This proliferation of data means that often, for any information query we might have, there are a variety of resources available that store data about the same domain and which are of varying quality. A distributed query engine needs to decide which of the many available resources to use in finding the solution to a query. In addition to finding the resources that have the required information, it may also be necessary to decide between different resources that have the same information available. In order for a distributed query engine to understand what information is available, the resources need to make descriptions of their contents available in a meaningful way. If the terms using in such a description are formally defined in an ontology, the query engine has access to the meaning of the terms in the description. This allows the query engine to make fully informed decisions about which resources are relevant to resolving a particular query.

There are a number of pragmatic issues in locating the resources that will be used to answer a query. For example, a particular user may - for whatever reason - prefer one resource over another as the source of some information.

Such personal preferences can be taken into account by the distributed query engine if a personal profile of a user's preferences is maintained. The query engine can make better informed decisions if the definitions of the terms used in such a profile are available to it in the form of a user *ontology*, which defines the terminology of a user or user-group.

2.4 Ontologies and Database Schemas

Ontologies and database schemas are closely related and people often have trouble deciding which is which. There is often no tangible difference, no way of identifying which representation is a schema and which is an ontology. This is especially true for schemas represented using a semantic data model. The main difference is one of purpose. An ontology is developed in order to define the meaning of the terms used in some domain whereas a schema is developed in order to model some data. Although there is often some correspondence between a data model and the meaning of the terms used, this is not necessarily the case. Both schemas and ontologies play key roles in heterogeneous information integration because both semantics and data structures are important.

For example, the terminology used in schemas is often not the best way to describe the content of a resource to people or machines. If we use the terms defined in a resource ontology to describe the contents of a resource, queries that are sent to the resource will also use these terms. In order to answer such queries, there needs to be a relationship defined between the ontology and the resource schema. Again, declarative mappings that can be interpreted by some mediator system are useful here. The structural information provided by schemas will enable the construction of executable queries such as SQL queries.

This is related to the discussion earlier about XML, where a database schema is analogous to an XML schema or DTD. As pointed out above, using XML is insufficient for determining the semantics of resources. A schema, whether specified using XML or some database schema language, needs an associated formal ontology in order to make the semantics of the resource clear. When the meaning of data and schemas is made explicit using an ontology, programs can be designed that exploit those semantics.

2.5 Entity Correspondence

Ontologies are used in e-commerce environments where data is scattered across heterogeneous distributed systems. In order for the consumer to have access to the maximum amount of available information, we want to be able to retrieve information from various systems and to integrate it. For example, we might want to integrate information from a supplier's product catalogue with customer reviews produced independently.

To gather all the information relevant to an entities, the correspondence between entities across resources must be established. For example, the academic records and criminal records of a person are likely to be stored in separated data resources. However, the way in which different resources identify individuals varies. For example, in relational databases entities are identified using key attributes. There is no guarantee that different relational databases use the same key attributes. Even when the same key attribute is used, different terms may be used to denote the attributes. How our systems can determine whether entities from different resources are the same or not is crucial to fusing information. Standard schemas do not provide a full solution here since many systems (*e.g.* KBSs, object-oriented databases) often do not have key attributes at all.

3. DOME Overview

The DOME project has been researching and developing ontology-based techniques to support the building of a "one-stop knowledge shop" for corporate information. We have developed a methodology, a set of tools and an architecture to enable enterprise-wide information management for data re-use and knowledge sharing. The system retrieves information from multiple resources to answer user queries and presents the results in a consistent way that is meaningful to the user. This section gives an overview of the DOME prototype system and some implementation details. Further details of DOME can be found in [Cui *et al.*, 2001]. Figure 1 shows the architecture of the DOME prototype.

The DOME prototype consists of a number of interacting components: an ontology server which is responsible for managing the definitions of terms, a mapping server which manages the relationships between ontologies, an engineering client with tools for developing and administrating a DOME system, a user client to support querying the knowledge shop, and a query engine for decomposing queries fusing the results to sub-queries. The prototype is implemented as an Enterprise JavaBean which provides two APIs - one for developers and one for users and applications.

3.1 Engineering client

A developer who wishes to set up a DOME system interacts with an engineering client which provides support in the development of the knowledge shop. This includes tools for the semi-automated extraction of ontologies from legacy systems [Yang *et al.*, 1999], for defining ontologies, for defining mappings between ontologies and between resource ontologies and database schemas.

We have developed a methodology that combines top-down and bottom-up ontology development approaches. This allows the engineer to select the best approach to take in developing an ontology. The top-down process starts with domain analysis to identify key concepts by consulting corporate data standards, information models, or generic ontologies such as Cyc or WordNet. Following that, the engineer defines competency questions [Gruninger and Fox, 1995.] The top down process results in the shared ontologies

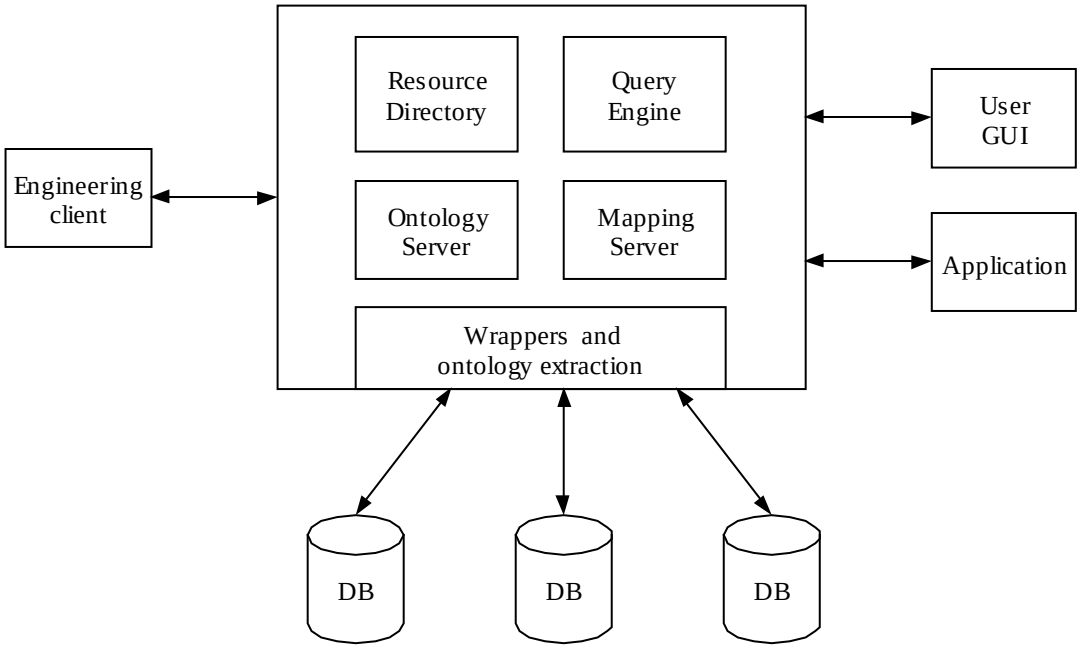


Figure 1: The DOME architecture

mentioned above. The bottom-up process starts with the underlying data sources. The extraction tool is applied to database schemas and application programs to produce initial ontologies which are further refined to become resource ontologies. We also provide for the development of *application ontologies*, which define the terminology of a user-group or client application. Application ontologies are defined by specialising the definitions in a shared ontology. Once the ontologies have been defined, they are stored in the ontology server.

The engineer also needs to define mappings between the resource ontologies and the shared ontology for a particular application. The rest of the ontology engineering task is to define mappings between the resource and shared ontologies using ontology mappings. Although we do not infer the mappings automatically, we can utilise ontologies to check the mappings for consistency. The engineer also needs to define mappings between the database schemas and the resource ontologies.

3.2 Ontology server

The ontology server stores the ontologies that are defined using the engineering client and allows access to the three kinds of ontologies in a DOME network: *shared*, *resource* and *application* ontologies. Shared ontologies contain definitions of general terms that are common across and between enterprises. A resource ontology contains definitions of terms used by a particular resource. These ontologies are stored in the DOME ontology server which implements ontologies using the description logic CLASSIC [Brachman *et al.*, 1992]. CLASSIC is used to both store ontologies and to make inferences. Access to the ontology server is through Open Knowledge Base Connectivity (OKBC) interface, which is a *de facto* standard for accessing knowledge bases [Chaudhri *et al.*, 1998].

3.3 User client

We provide users with tools to access the knowledge shop. We have defined a simple API that allows a user client or an application to querying the distributed information space. The user client also provides facilities for loading and browsing specific ontologies in the knowledge shop to view what is available in the whole information space. Queries are passed to DOME as strings which conform to an XML schema which defines the syntax of the DOME query language. This is similar to SQL but doesn't require that we specify on which attributes to make joins between concepts since this will be identified automatically by the query engine. Queries are formed using the terminology defined in an application ontology and the results which are returned are represented using the same terminology, hence hiding the details of the different systems, their distribution, structure, syntax or semantics from the user.

3.4 Mapping Server

The mapping server stores the mappings between ontologies which are defined by the engineer in setting up a DOME network. The mapping server also stores generic conversion functions which can be utilised by the engineer when defining a mapping from one ontology to another. These

mappings are specified using a declarative syntax, which allows the mappings to be straightforwardly modified and reused. The query engine queries the mapping server when it needs to translate between ontologies in solving a query.

3.5 Wrappers

Most interaction between a resource and the DOME network occurs via wrappers. A wrapper performs translations of queries expressed in the DOME query syntax and terminology of the resource ontology to queries expressed in the syntax of the resource query language and the terminology of the resource schema. They also perform any translations required to put the results into the terminology of the resource ontology. Although they are configured for particular resources, DOME wrappers are generic across resources of the same type *e.g.* wrappers of SQL databases utilise the same code.

3.6 Resource Directory

When a resource is connected to a DOME network, its wrapper will inform the DOME directory about its existence and pass to the resource directory a description of the contents of the resource, expressed in terms of the relevant resource ontology. This ensures that the query engine is able to identify what information is available without having to access the schema of the resource. When a wrapper is - for whatever reason - no longer able to provide information from a resource, it will inform the resource directory which is then able to discount that resource from any future query solving.

3.7 Query engine

Upon receiving a query, the DOME query engine first needs to decide which resources are relevant to that query. It obtains a list of currently available and relevant resources by consulting the directory. Based on this information, the query engine decomposes the query into sub-queries. The query engine ensures that the decomposition is performed in such a way that the results to the sub-queries, once they are received from the resources, can then be integrated. It then translates queries from the ontology of the query to that of the relevant resource and will send the sub-queries to the resources. Once the results are received, the query engine will integrate the results.

4. DOME Demonstrator

We have developed a demonstrator for the DOME prototype based on a database marketing scenario. Database marketing involves targeting marketing information from customer information stored in databases. Typically, queries are *ad hoc*, that is, it is difficult to pre-define a set of typical queries. Also, customer information is necessarily split across many different databases *e.g.* a customer may have multiple products, the records for which are stored in different databases. This requires that queries often need to join information from a wide variety of databases. As the databases we used are developed independently and serve different applications, DOME has to search for resources which hold data about customers that it is possible to integrate. The resources that are used varies from query to query. The databases also have different levels of data quality - there are incorrect entries, missing records, etc. As DOME allows mappings to be specified between the shared and resource ontologies, we have some control over which resources are utilised for data that is available from multiple databases. By only defining mappings between the shared ontology and the parts of the resource ontology for which the resource is a trusted sources of information, we can limit the parts of a resource that is used to solve queries.

5. Conclusions

Semantic interoperation is one of the main obstacles to free and full electronic commerce. Understanding what is available is a necessary prerequisite to a successful business transaction. We have described a number of issues involved in supporting the interoperation of computer systems at the semantic level. We have also described the architecture of the DOME system that we have developed to illustrate our approach to overcoming some of these problems. We believe that the proof-of-concept demonstrator we have developed supports the utility of ontologies in integrating heterogeneous information resources for applications such as e-commerce. DOME provides functionality to (i) support a system engineer in providing an integrated view of networked heterogeneous databases, (ii) allow a user to select and browse definitions of terminologies and to pose queries in their chosen vocabulary and (iii) answer user queries based on the information available. This work is ongoing and there are a number of areas currently being explored. For example, an increasing number of resources that use some form of XML technology are becoming available and we are currently developing components that will allow data retrieved from such resources to be integrated with data retrieved from other kinds of resources such as relational databases. We believe strongly that even in a world where there are many such resources, there will still be a role for formal ontologies in enabling semantic interoperability.

References

[Brachman *et al.*, 1992] R.J. Brachman, A. Borgida, D.L. McGuinness, P.F. Patel-Schneider and L. Alperin Resnick. The CLASSIC Knowledge Representation System, or KL-ONE: The Next Generation. *Proceedings of the 1992 International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, June 1992.

[Chaudhri *et al.*, 1998] V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, and J.P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. *Proceedings of AAAI-98*, pages 600-607, Madison, WI, 1998

[Cui *et al.*, 2001] Z. Cui, M.D.J. Cox and D.M. Jones. An Environment for Managing Enterprise Domain Ontologies. M. Rossi and K. Siau (eds.) *Information Modelling in the New Millennium*, Idea Group Publishing, London.

[Gruninger and Fox, 1995] M. Gruninger, and M.S. Fox. Methodology for the Design and Evaluation of Ontologies. *IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 1995

[Jones *et al.*, 1998] D.M. Jones, T.J.M. Bench-Capon and P.R.S. Visser. Methodologies for Ontology Development. *Proceedings IT&KNOWS Conference of the 15th IFIP World Computer Congress*, Budapest, Chapman-Hall.

[Sheth, 1998] A.P. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.) *Interoperating Geographic Information Systems*, Kluwer.

[Visser *et al.*, 1998] P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon and M.J.R. Shave. Assessing Heterogeneity by Classifying Ontology Mismatches. *Proceedings International Conference on Formal Ontology in Information Systems - FOIS'98*, IOS Press.

[Yang *et al.*, 1999] H. Yang, Z. Cui and P.D. O'Brien. Extracting Ontologies from Legacy Systems for Understanding and Re-engineering. *Proceedings of 23rd IEEE International Conference on Computer Software and Applications*, Pheonix, AZ, October 1999.

Extending RDF(S) with Contextual and Definitional Knowledge

Alexandre Delteil, Catherine Faron-Zucker
ACACIA project, INRIA, 2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis, France
{Alexandre.Delteil, Catherine.Faron}@sophia.inria.fr

Abstract

RDF(S) is the emerging standard for knowledge representation on the Web. In the European IST project CoMMA dedicated to ontology guided information retrieval in a corporate memory, the semantic annotations describing the Intranet documents are represented in RDF(S). In this context, the RDF(S) expressivity appears to be too much limited. Compared to object-oriented representation languages, description logics, or conceptual graphs, RDF(S) does not enable to define classes or properties nor represent axioms inside an ontology. In this paper, we propose an extension of RDF(S) to express this kind of definitional knowledge, and more generally contextual knowledge on the Semantic Web. We hope that DRDF(S) will contribute to the ongoing work of the W3C committee for improving RDFS and meet the needs of the e-business community.

1 Introduction

The need of a Semantic Web is now well recognized and always more emphasized [Berners Lee, 1999]. The huge amount of information available on the web has become overwhelming, and knowledge based reasoning now is the key to lead the Web to its full potential. In the last few years, a new generation of knowledge based search engines has arisen, among which the most famous are *SHOE* [Luke *et al.*, 1997] and *Ontobroker* [Fensel *et al.*, 1998]. They rely on extensions of HTML to annotate Web documents with semantic metadata, thus enabling semantic content guided search. For interoperability on the Web, the importance of widely accepted standards is emphasized. *Resource Description Framework* (RDF) is the emerging standard proposed by the W3C for the representation and exchange of metadata on the Semantic Web [RDF, 1999]; it has an XML syntax. *RDF Schema* (RDFS) is the standard dedicated to the representation of ontological knowledge used in RDF statements [RDFS, 2000].

In the context of the 'CoMMA' European IST project, RDFS is the knowledge representation language used to annotate the Intranet documents of an organization. These annotations are exploited for knowledge based information retrieval on the Intranet by using the inference engine *CORESE* implemented in our team [Corby *et al.*, 2000]. However the expressivity of RDF(S) appears too much limited to represent the ontological knowledge of the corporate memory. Inference rules representing domain axioms, class and property definitions are crucial for intelligent information retrieval on the Web. The need for inference rules is well-known since the first information retrieval systems on the Semantic Web. Axiomatic knowledge, algebraic properties of relations, or domain axioms are the key to discover implicit knowledge in Web page annotations so that information retrieval be independent of the point of view adopted when annotating [Heflin *et al.*, 1998]. [Martin *et al.*, 2000] claim the need for additional features and conventions in RDF.

When compared to object-oriented knowledge representation languages, description logics, or conceptual graphs, RDF(S) does not enable to define classes or properties nor represent axioms [DAML, 2001; OIL, 2000]. In this paper, we propose an extension of RDF(S) with class, property and axiom definitions. We call it DRDF(S) for *Defined Resource Description Framework*. DRDFS more generally enables to express contextual knowledge on the Web. The RDF philosophy consists in letting anybody free to declare anything about any resource. Therefore the knowledge of who and in which context a special annotation has been stated is crucial. DRDF(S) enables to assign a context to any cluster of annotations, in particular for definitional contexts. We hope that DRDF(S) will contribute to the ongoing work of the W3C committee for improving RDFS and meet the needs of the e-business community.

In the next section, we present the RDF(S) model. Section 3 is dedicated to the comparison of RDF(S) and the Conceptual Graphs model. Section 4 presents an extension of RDF(S) with contexts, and section 5 an extension of existential quantification handling. The RDF extensions for defining classes, properties and axioms are presented in sections 6, 7 and 8. The metamodel of DRDF(S) is described in section 9. Section 10 is dedicated to a comparison between DRDF(S) and other Web languages.

2 The RDF(S) Model

2.1 RDF and RDFS

RDF is the emerging Web standard for annotating resources, such as images or documents, with semantic metadata [RDF, 1999]. These Web resources are identified by their URIs. In addition, anonymous resources provide a limited way of existential quantification. An RDF description consists in a set of statements; each one specifying a value of a property of a resource. A statement is thus a triple (resource, property, value), a value being either a resource or a literal. The RDF data model is close to semantic nets. A set of statements is viewed as a directed labeled graph: a vertex is either a resource or a literal; an arc between two vertices is labeled by a property. RDF is provided with an XML syntax.

Figure 1 presents an example of RDF graph and its XML serialization. It is the annotation of the Web page of T-Nova which is a subdivision of Deutsche Telekom. The examples highlighting our paper are all based on the CoMMA ontology.

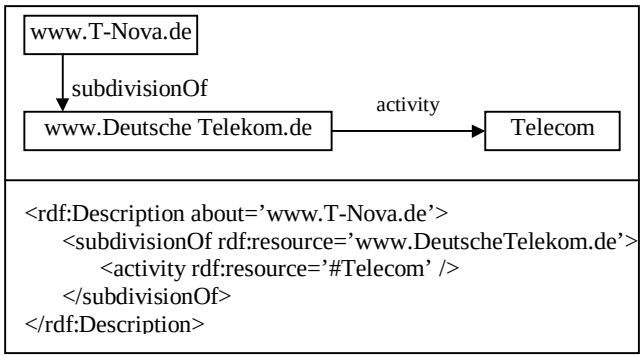


Figure 1. An example of RDF annotation.

RDF Schema (RDFS) is dedicated to the specification of schemas representing the ontological knowledge used in RDF statements [RDFS, 2000]. A schema consists in a set of declarations of classes and properties. Multi-inheritance is allowed for both classes and properties. A property is declared with a signature allowing several domains and one single range: the domains of a property constraint the classes this property can be applied to, and its range the class the value of this property belongs to.

The RDFS metamodel is presented in Figure 2. This definition is recursive: the terms of RDFS are themselves defined in the RDFS model. More precisely, the RDFS metamodel itself is defined as a set of statements by using the two core RDFS properties: subclassOf and type which denote respectively the subsumption relation between classes and the instantiation relation between an instance and a class.

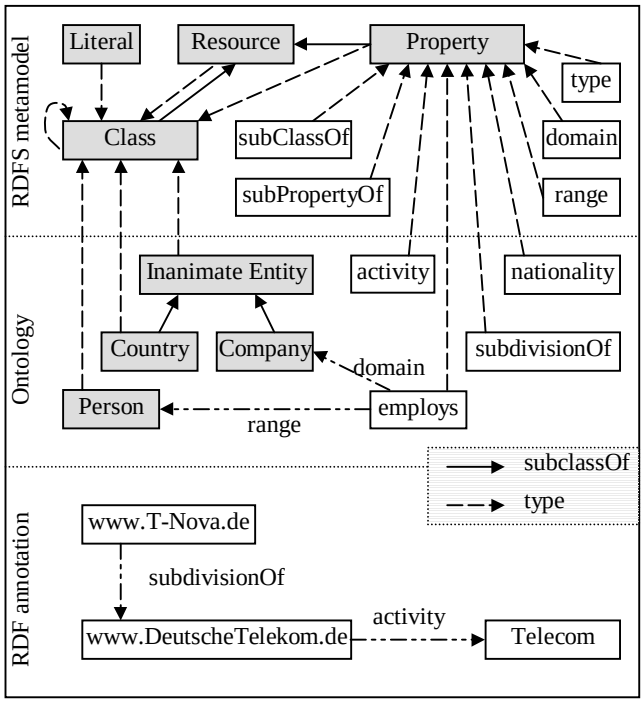


Figure 2. The RDFS metamodel and an RDFS schema.

To represent domain specific knowledge, a schema is defined by refining the core RDFS. As shown in Figure 2, domain specific classes are declared as instances of the "Class" resource and domain specific properties are declared as instances of the "Property" resource. The "subclassOf" and "subPropertyOf" properties enable to define class hierarchies and property hierarchies.

2.2 RDF Limitations

A Triple Model. The RDF data model is a triple model: an RDF statement is a triple (resource, property, value). When asserted, RDF triples are clustered inside annotations. An annotation can thus be viewed as a graph, subgraph of the great RDF graph representing the whole set of annotations on the Web. However, "there is no distinction between the statements made in a single sentence and the statements made in separate sentences" [RDF, 1999]. Let us consider two different annotations relative to two different research projects which the employee 46 of T-Nova participates to:

- {(employee-46, worksIn, T-Nova), (employee-46, project, CoMMA), (employee-46, activity, endUser)}
- {(employee-46, worksIn, T-Nova), (employee-46, project, projectX), (employee-46, activity, developer)}.

The whole RDF graph does not distinguish between these two clusters of statements. Employee 46 is both endUser and developer: the knowledge of which activity inside of a project he is implicated in is lost.

RDF Reification. The RDF model is provided with a reification mechanism dedicated to higher order statements about statements. A statement (r, p, v) is reified into a resource s described by the four following properties: the *subject* property identifies the resource r, the *predicate* property identifies the original property p, the *object* property identifies the property value v, the *type* property describes the type of s; all reified statements are instances of *Statement*. Figure 3 presents the following reification: 'Observer-3002 says that the rating of Newsletter-425 is seminal'.

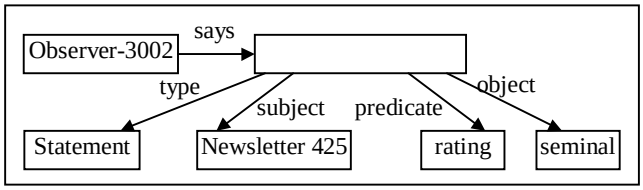
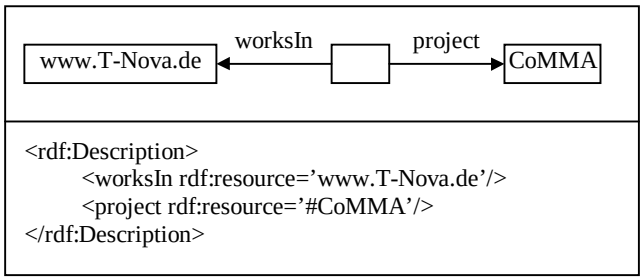


Figure 3. An example of reification.

Let us consider now the reification of a set of statements. It requires the use of a container to refer to the collection of the resources reifying these statements. This leads to quite complicate graphs (see Figure 10 in [RDF, 1999]). Moreover a statement containing an anonymous resource can not always be reified: the values of the properties *subject* and *object* must have an identifier.

Existential quantification. The RDF model focuses on the description of identified resources but allows a limited form of existential quantification through the anonymous resource feature. Let us consider the following RDF statements describing an anonymous resource:




```
<rdf:Description>
      <worksIn rdf:resource='www.T-Nova.de'/>
      <project rdf:resource='#CoMMA'/>
</rdf:Description>
```

Figure 4. An example of anonymous resource.

This existential quantification is handled by automatically generating an ID for the anonymous resource. However, such a handling of existential knowledge through constants is a limited solution and a graph containing a cycle with more than one anonymous resource can not be represented in RDF (Figure 5).

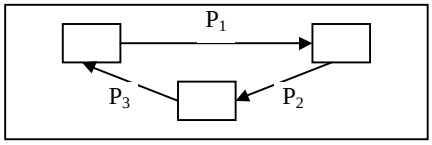


Figure 5. An RDF graph without XML serialization

Classes and properties. An RDF Schema is made of atomic classes and properties. The RDFS model does not enable the definition of classes or properties. More generally, inferences cannot be represented in the model.

3 The RDF(S) and Conceptual Graphs Models

3.1 The Conceptual Graphs Model

Conceptual Graphs [Sowa, 1984; Sowa, 1999] is a knowledge representation model descending from Existential Graphs [Pierce, 1932] and Semantic Networks. A conceptual graph is a bipartite (not necessarily connected) graph composed of concept nodes, and relation nodes describing relations between these concepts.

Each concept node c of a graph G is labeled by a couple $<type(c)\ ,\ referent(c)>$, where $referent(c)$ is either the generic marker * corresponding to the existential quantification or an individual marker corresponding to an identifier; M is the set of all the individual markers. Each relation node r of a graph G is labeled by a relation type $type(r)$; each relation type is associated with a signature expressing constraints on the types of the concepts that may be linked to its arcs in a graph.

Concept types (respectively relation types of same arity) build up a set T_c (resp. T_r) partially ordered by a generalization/specialization relation $\leq_c$ (resp. $\geq_r$). (T_c, T_r, M) defines the *support* upon which conceptual graphs are constructed. A support thus represents a domain ontology.
The semantics of the Conceptual Graphs model relies on the translation of a graph G into a first order logic formula thanks to a Φ operator as defined in [Sowa, 1984]: $\Phi(G)$ is the conjunction of unary predicates translating the concept nodes of G and n-ary predicates translating the n-ary relation nodes of G; an existential quantification is introduced for each generic concept.

Conceptual graphs are provided with a generalization/specialization relation $\leq_G$ corresponding to the logical implication: $G_1 \leq_G G_2$ iff $\Phi(G_1) => \Phi(G_2)$. The fundamental operation called *projection* enables to determine the generalization relation between two graphs: $G_1 \leq_G G_2$ iff there exists a projection π from G_2 to G_1. π is a graph morphism such that the label of a node n_1 of G_1 is a specialization of the label of a node n_2 of G_2 with $n_1 = \pi(n_2)$. Reasoning with conceptual graphs is based on the projection, which is sound and complete with respect to logical deduction.

3.2 Mapping of the RDF(S) and CG models

The RDFS and CG models share many common features and a mapping can easily be established between RDFS and a large subset of the CG model. An in-depth comparison of both models is studied in [Corby *et al.*, 2000].

- Both models distinguish between ontological knowledge and assertional knowledge. First the class (resp. property) hierarchy in a RDF Schema corresponds to the concept (resp. relation) type hierarchy in a CG support; this distinction is common to most knowledge representation languages. Second, and more important, RDFS properties are declared as first class entities like RDFS classes, in just the same way that relation types are declared independently of concept types. This is this common handling of properties that makes relevant the mapping of RDFS and CG models. In particular, it can be opposed to object-oriented approaches, where properties are defined inside of classes.

- In both models, the assertional knowledge is positive, conjunctive and existential.

- Both models allow a way of reification.

- In both models, the assertional knowledge is represented by directed labeled graphs. An RDF graph G may be translated into a conceptual graph CG as follows:

  - Each arc labeled with a property p in G is translated into a relation node of type p in CG.
  - Each node labeled with an identified resource in G is translated into an individual concept in CG whose marker is the resource identifier. Its type corresponds to the class the identified resource is linked to by a *rdf:type* property in G.
  - Each node labeled with an anonymous resource in G is translated into a generic concept in CG. Its type corresponds to the class the anonymous resource is linked to by a *rdf:type* property in G.

Regarding the handling of classes and properties, the RDF(S) and CG models differ on several points. However these differences can be quite easily handled when mapping RDF and CG models.

- RDF binary properties versus CG n-ary relation types: the RDF data model intrinsically only supports binary relations, whereas the CG model authorizes n-ary relations. However it is possible to express n-ary relations with binary properties by using an intermediate resource with additional properties of this resource giving the remaining relations [RDF, 1999].

- RDF multi-instantiation versus CG mono-instantiation: the RDF data model supports multi-instantiation whereas the CG model does not. However the declaration of a resource as instance of several classes in RDF can be translated in the CG model by generating the concept type corresponding to the most general specialization of the concept types translating these classes.

- Property and relation type signatures: in the RDF data model, a property may have several domains whereas in the CG model, a relation type is constrained by a single domain. However the multiple domains of an RDF property may be translated into a single domain of a CG relation type by generating the concept type corresponding to the most general specialization of the concept types translating the domains of the property.

3.3Additional expressivity of the CG model

In addition to the features the CG model shares with RDF(S), it is provided with additional features insuring a greater expressivity. Regarding the existing mapping between both models, these features will be the key to an extension of RDF(S) based on the CG model [Delteil *et al.*, 2001].

A graph model

A conceptual graph represents a piece of knowledge separate from the other conceptual graphs of the base it belongs to. Let us consider again the two projects of T-Nova which Employee-46 participates in. The statements relative to one project are clustered in one conceptual graph and then separated from the statements relative to the other projects.
The two conceptual graphs are the following:

- [Project : CoMMA] <--(project)<-- [T : Employee-46] -->(activity)--> [EndUser: *].

- [Project : projectX] <--(project)<-- [T : Employee-46] -->(activity)--> [Developer : *].

A CG base is a set of conceptual graphs that cannot be decomposed in smaller pieces of knowledge without loss of information.

Reification

A conceptual graph g is reified into a marker whose value is g.
Let us consider again the following reification: 'Observer-3002 says that the rating of Newsletter-425 is seminal'. It is represented by the following conceptual graph:

[T : Observer-3002] ---> (says) ---> [Proposition :
[T : Newsletter-425] ---> (rating) ---> [T : seminal]].

In the RDF model, the reification of a set of statements requires the use of a container to refer to the collection of the resources reifying these statements. In the CG model, since the notion of graph is intrinsic to the model, the

equivalent reification remains based on the initial basic mechanism.

Existential Quantification
The CG model allows to represent every existential, positive and conjunctive proposition without any restriction.

Type definitions and axioms
In the CG model, concept type and relation type are either atomic or defined [Leclere, 1997]. Graph rules allow the representation of inference rules [Salvat and Mugnier, 1996].

Starting from the correspondence between RDF(S) and the conceptual graph model, we propose an extension of RDF(S) based on the CG model to provide the former with an expressivity equivalent to the one of the latter. We call this extension DRDFS.

4 Extending RDFS with contexts

The RDF model provides no way of expressing independent pieces of knowledge. We propose to extend RDF with a notion of context to express the clustering of statements much more easily than RDF containers. A context identifies a sub-graph of the whole RDF graph, so that a triple can be stated inside of a special context. This extension is based on the similarities between the RDF and CG models: a context is just the translation of a conceptual graph. The CG model provides a direct way of expressing independent pieces of knowledge through graphs: a conceptual graph implicitly defines a context. The representation of contexts for various applications (quotations, viewpoint, …) is direct in the CG model. Conceptual graphs are particularly useful as definitional contexts enabling the definition of concepts or axioms. By introducing contexts in RDF, we propose a very general mechanism that will be the keystone of further extensions, like class or rule definitions.

To extend RDFS with contexts, we introduce the following new RDF primitives:
'Context': A context is a resource of type *Context*. *Context* is a subclass of *rdfs:Class*.
'isContextOf': A resource is linked by a *isContextOf* property to the context it belongs to.
'referent': An anonymous resource is linked by a *referent* property to the identified resource it refers to.

The rules for constructing RDF contexts are based on the translation of conceptual graphs into RDF:

- An individual concept [C: r] of a conceptual graph G is represented by three RDF triples (c∅, type, C), (c∅, referent, r), (G, isContextOf, c∅), where c∅ is an anonymous resource (whose ID is automatically generated by RDF parsers).
- A generic concept [C: *] of a conceptual graph G is represented by two RDF triples (c∅, type, C), (G, isContextOf, c∅).
- A generic concept [C: *x] of a graph G is represented by three RDF triples (c∅, type, C), (c∅, referent, x), (G, isContextOf, c∅), where x is an instance of the class *Variable* (this class will be further described in next section).
- A relation R between two concepts [C_1: r_1] and [C_2: r_2] of a conceptual graph G is represented by an RDF property P between the two anonymous resources $c\emptyset_1$ and $c\emptyset_2$.
- The resource G is an instance of the class Context; this is represented by the triple (G, type, Context).

Note that to represent a context, it could be sufficient to link a single anonymous resource of it to the resource G representing it by the *isContextOf* property.

Let us consider again the two projects of T-Nova which Employee-46 participates in. As shown in Figure 6, the statements relative to one project can now be clustered in a context and then separated from the statements relative to the other projects.

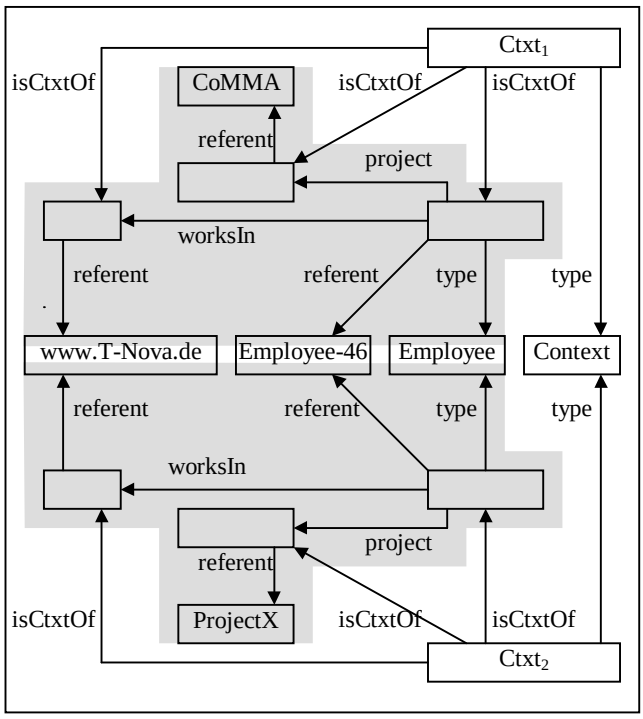


Figure 6. Two contexts about the resource Employee-46.

The rules for extracting the set S of the triples belonging to a context from the whole RDF graph are the following:

- Select a resource G of type Context; S <-- {(G, type, Context)}.
- Select all the anonymous resources $c\varnothing_i$ for which the value of the *isContextOf* property is G; for each i, S <-- S ∪ {(G, isContextOf, $c\varnothing_i$)}.
- Select all the identified resources r_j values of a *referent* property of a resource $c\varnothing_i$; S <-- S ∪ {($c\varnothing_i$, referent, r_j)}.
- Select all the properties p_{ik} between two resources $c\varnothing_i$ and $c\varnothing_k$; S <-- S ∪ {($c\varnothing_i$, p_{ik}, $c\varnothing_k$)}.

Regarding the whole RDF graph, a context defines, just like a conceptual graph, a piece of knowledge, i.e. an independent clustering of statements. A context is defined from a resource G of type Context as the largest subgraph of the whole RDF graph whose all internal nodes excepted G are anonymous resources $c\varnothing_i$. A context is thus an abstraction that enables to talk about representations of resources (through anonymous resources) rather than directly about resources. For instance, in Figure 6, the resource Empoyee-46 is referred to by two distinct anonymous resources in two different contexts. Anonymous resources are "externally identified" by the *referent* property.
This general notion of context will appear of particular interest for expressing definitional contexts.

5 Extension of RDF(S) with existential quantification

The RDF model allows a limited form of existential quantification through the anonymous resource feature. The introduction of the *referent* property provides the RDF model with a general mechanism for existential quantification handling.

To extend RDFS with existential quantification, we introduce the following new RDF primitives:
'Variable': A variable is a resource of type *Variable*. *Variable* is a subclass of *rdfs:Class*.
'parameter': A variable is linked by a *parameter* property to the context it belongs to.

An existential quantification is represented by an anonymous resource described by a *referent* property whose value is an instance of *Variable*. The scope of a variable is the context it belongs to, just like in first-order logic, where the scope of a variable is the formula it belongs to.

In an RDF graph, an anonymous resource can be duplicated into several anonymous resources coreferencing a same variable; the new graph remains semantically equivalent to the initial one. This enables the XML serialization of RDF graphs embedding a cycle with anonymous resources. Figure 7 presents one DRDF graph semantically equivalent to the RDF graph of Figure 5 that could not be serialized in the XML syntax. The cycle is resolved by introducing a second anonymous resource and two *referent* properties sharing the same value:

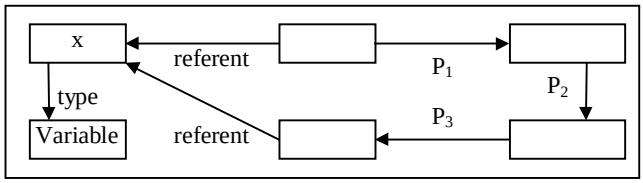


Figure 7. An example of existential quantification.

6 Extending RDFS with class definition

DRDF(S) class definition is descended from type definition in the CG model. A class definition is a monadic abstraction, i.e. a context whose one resource of type Variable is considered as formal parameter.

To extend RDFS with class definitions, we introduce the following new RDF primitives:
'DefinedClass': A defined class is of type *DefinedClass*. *DefinedClass* is a subclass of *rdfs:Class*.
'hasDefinition': A defined class is linked by a *hasDefinition* property to its definitional context.
'formalParameter': The variable linked to the definitional context by a *FormalParameter* property corresponds to the formal parameter of a monadic lambda abstraction.

Figure 8 describes the definition of the 'WebPage' class, as a document having HTML for representation system. The XML serialization of this graph is provided in appendix 2.

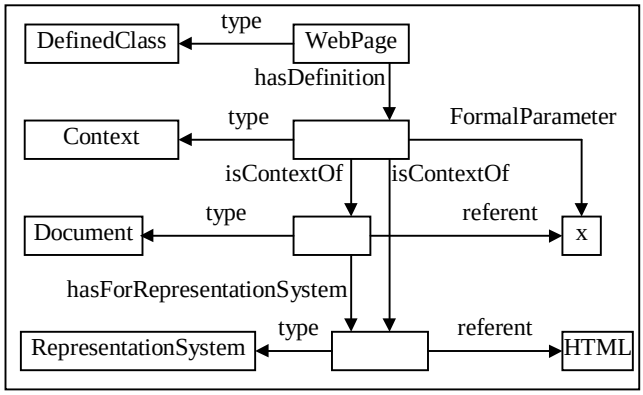


Figure 8. Definition of the 'WebPage' class.

7 Extending RDFS with property definition

DRDF(S) property definition is descended from type definition in the CG model. A property definition is a diadic abstraction, i.e. a context whose two resources of type Variable are considered as formal parameters.

To extend RDFS with property definitions, we introduce the following new RDF primitives:
'DefinedProperty': A defined property is of type *DefinedProperty*. *DefinedProperty* is a subclass of *rdf:Property*.
'firstFormalParameter' and 'secondFormalParameter': The variables linked to the definitional context by these properties correspond to the formal parameters of a diadic lambda abstraction.

Figure 9 describes the definition of the 'colleague' property, as a relation between two persons working in the same institute.

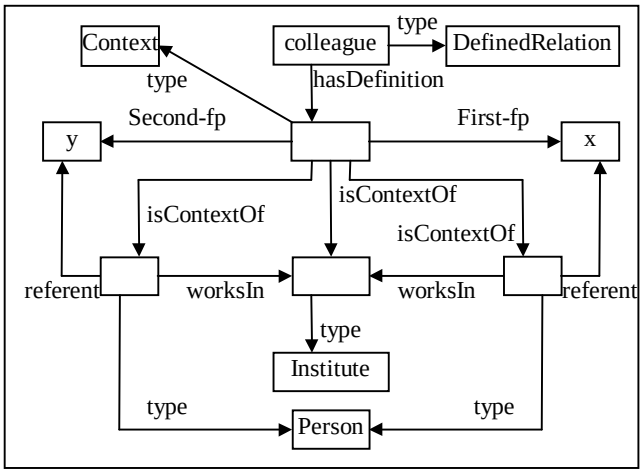


Figure 9. Definition of the 'colleague' property.

8 Extending RDFS with axioms

DRDF(S) axiom definition is descended from graph rules in the CG model. An axiom is a couple of lambda abstractions, i.e. two contexts representing the hypothesis and the conclusion.

To extend RDFS with axiom definitions, we introduce the following new RDF primitives:
'Axiom': An axiom is a resource of type *Axiom*. *Axiom* is a subclass of *Context*.
'if': An axiom is linked by an *if* property to the context defining its hypothesis.
'then': An axiom is linked by a *then* property to the context defining its conclusion.

The variables linked by a *formalParameter* property to the resource of type *Axiom* correspond to the formal parameters common to the two lambda abstractions.
Figure 10 describes the definition of the axiom "If x is colleague of y, then y is colleague of x".

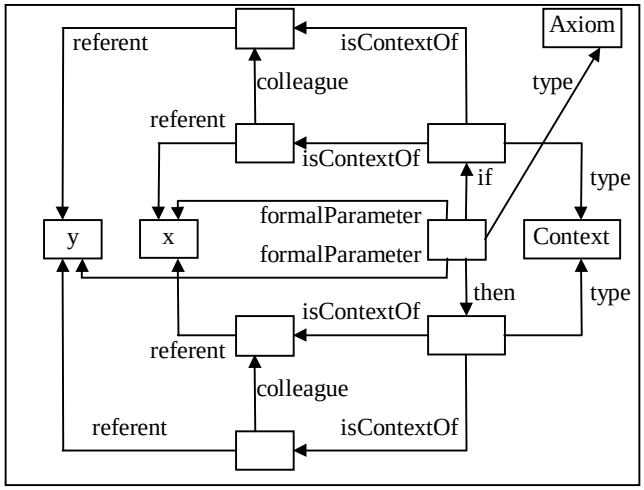


Figure 10. Definition of an axiom.

9 The Defined Resource Description Framework Schema (DRDFS)

9.1 The RDFS metamodel

The extensions introduced in previous sections are a refinement of the core RDFS and remain totally compliant with the RDF triple model. We call Defined Resource Description Framework Schema (DRDFS) the set of RDFS primitives augmented with the ones we introduce. The namespace prefix 'drdfs' is used to differentiate these new elements from the standard RDFS ones. For readability, we respect the RDFS convention that the first letter of class names is capital while the first letter of property names is small.
The metamodel of DRDFS is presented in Figure 11; its XML serialization is provided in Appendix 1.

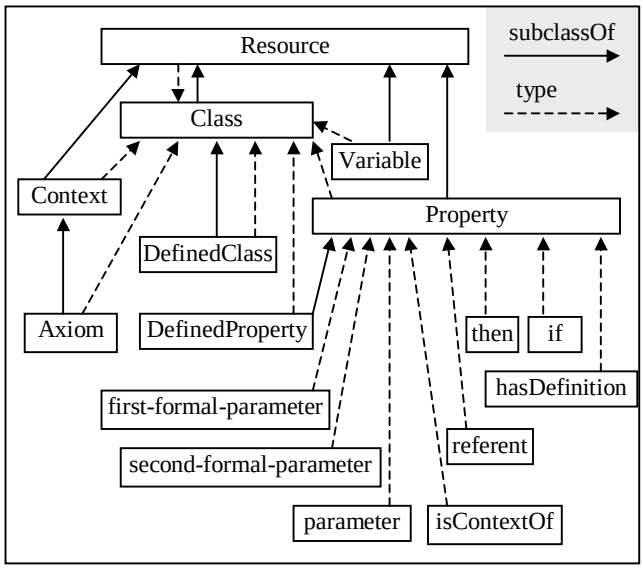


Figure 11. The DRDFS metamodel.

9.2Semantics of DRDFS

The semantics of DRDFS relies on its translation into the CG formalism. Conceptual graphs are themselves translated into first order logic formulae thanks to the Φ operator defined in [Sowa, 1984].

9.3Reasoning with DRDFS

The RDF model is dedicated to knowledge representation and interoperability on the Web. It does not address the problem of reasoning with the formalized knowledge; the only inference mechanisms it provides are the subsumption relations between classes and properties. Regarding the mapping established between the RDF and CG models in [Corby *et al.*, 2000], CG engines are good candidates for reasoning on the Semantic Web: the CORESE system developed in our team is a first step in this direction. Algorithms will be implemented in CORESE for reasoning with type definitions. They will be based on [Leclere, 1997] and will enable reasoning with the full DRDFS.

10 Related Work

Several languages for ontology representation and exchange are existing [Corcho and Gomez-Perez, 2000], among which RDF(S), OIL [Fensel *et al.*, 2000] and DAML [DAML, 2001] are dedicated to the Semantic Web. Like DRDF(S), OIL and DAML are tentatives of improvement of RDF(S); they are defined as an RDF Schema.

OIL enables to define classes and restrict property ranges and domains through boolean combinations of classes. In particular, it enables negation in class definitions, which is not provided in DRDFS. OIL is based on a DL. When compared to it, what DRDFS provides with its CG's expressivity is the possibility to express any positive, conjunctive and existential graph in a definition. The absence of variables in DLs does not enable to express RDF graphs embedding cycles; the class definitions in OIL are then limited to 'serializable' graphs. Contrary to OIL, DRDFS stays in the spirit of RDF(S), namely the representation of positive, conjunctive and existential knowledge. In our opinion, this better meets the needs of the Semantic Web.

DAML provides primitives to express relations between classes (disjonction, intersection, union, complementarity, ...) and enrich properties (minimal and maximal cardinality, transitivity, inverse, ...). DAML is provided with OOL features. It provides no mechanism for class or property definitions. It is therefore orthogonal to both OIL and DRDFS. As the merge of DAML and OIL led to DAML+OIL, it should be interesting to integrate the DAML features into DRDF(S).

In addition, DRDFS addresses the problem of the representation of contextual knowledge on the semantic web. This is of special interest to identify the origin of an annotation on the Web.

11 Conclusion

DRDF(S) is an extension of RDF(S) dedicated to ontology representation on the Semantic Web. It enables the representation of axioms, class and property definitions in ontologies. More generally, it provides a way to represent contextual knowledge on the Web.

In the framework of the CoMMA project, DRDF(S) should enable the representation of rich domain ontologies for intelligent IR in a company's Intranet. Since DRDF(S) is an RDF Schema, it is compliant with existing RDF parsers. However the semantics of the primitives specific to DRDF(S) can not be understood by them. We are currently working on a DRDF(S) interpreter for the existing platform CORESE.

The grounds of DRDF(S) rely on the existing mapping between RDF(S) and CGs; it is an extension of RDF(S) guided by the CG features. Regarding the similarities the RDF(S) and CG models share, the latter could contribute to the elaboration of a standard language for knowledge representation, interoperability and reasoning on the Semantic Web. We hope that DRDF(S) will contribute to the ongoing work of the W3C committee for improving RDFS and meet the needs of the e-business community.

References

[Berners Lee, 1999]. T. Berners Lee. Weaving the Web, Harper San Francisco.

[Chein and Mugnier, 1997] M. Chein, M.L. Mugnier. Positive Nested Conceptual Graphs. In proc. of the 5th International Conference on Conceptual Structures (ICCS'97), Seattle, WA, USA, Lukose D., Delugach, Keeler M. Searle L. and Sowa J. eds, Lecture Notes in Artificial Intelligence, LNIA 1257, Springer Verlag, p. 95-109, 1997.

[Corby *et al.*, 2000] O. Corby, R. Dieng, C. Hebert. A conceptual graph model for W3C Resource Description Framework. In proc. of the 8th International Conference on Conceptual Structures (ICCS'00), Darmstadt, Germany, Lecture Notes in Artificial Intelligence, LNCS 1867, Springer Verlag, 2000.

[Corcho and Gomez-Perez, 2000] O. Corcho, A. Gomez-Perez. A Roadmap to Ontology Specification Languages. In proc. of the 12th EKAW 2000, Juan-Les-Pins, France, LNAI 1937, Springer Verlag, p. 80-96, 2000.

[DAML, 2001] Darpa Agent Markup Language. http://www.daml.org.

[Delteil *et al.*, 2001] A. Delteil, C. Faron-Zucker and R. Dieng. Extensions of RDFS Based on the Conceptual Graph Model. To appear in proc. of ICCS'2001.

[Fensel *et al.*, 2000] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann and M. Klein. OIL in a nutshell. In proc of the 12th EKAW, Juan-Les-Pins, France, LNAI 1937, Springer Verlag, p. 1-16, 2000.

[Fensel *et al.*, 1998] D. Fensel, S. Decker, M. Erdmann, R. Studer. Ontobroker: Or How to Enable Intelligent Access to the WWW. In proc. of the 11th International Workshop

on Knowledge Acquisition, Modeling and Management (KAW 1998), Banff, Canada, 1998.
[Heflin *et al.*, 1998] J. Heflin, J. Hendler, S. Luke. Reading between the Lines: Using SHOE to Discover Implicit Knowledge from the Web. In proc. of the AAAI Workshop on Artificial Intelligence and Information Integration. WS-98-14. AAAI Press, p. 51-57, 1998.
[Leclere, 1997] M. Leclere. Reasoning with type definitions. In proc. of the 5th International Conference on Conceptual Structures (ICCS'97), Seattle, WA, USA, Lukose D., Delugach, Keeler M. Searle L. and Sowa J. eds, Lecture Notes in Artificial Intelligence, LNIA 1257, Springer Verlag, p. 401-415, 1997.
[Luke *et al.*, 1997] S. Luke, L. Spector, D. Rager, J. Hendler. Ontology based Web agents, In proc. of the 1st International Conference on Autonomous Agent, 1997.
[Martin and Eklund, 2000] P. Martin, P. Eklund. Conventions for Knowledge Representation via RDF, In proc. of WebNet 2000, San Antonio, Texas, ACCE press, p. 378-383, 2000.
[OIL, 2000] Ontology Inference Layer. http://www.ontoknowledge.org/oil.
[Pierce, 1932] C.S. Pierce. Collected Papers of Charles Sanders Pierce. In HARTSHORNE C. & WEISS P. eds. Harvard University Press, Cambridge, MA, 1932.
[RDF, 1999] Resource Description Framework Model and Syntax Specification, 1999. W3C Recommendation.
[RDFS, 2000] Resource Description Framework Schema Specification, 2000. W3C Candidate Recommendation.
[Salvat and Mugnier, 1996] E. Salvat, M.L. Mugnier. Sound and complete forward and backward chainings of graph rules, In proc. of the 4th International Conference on Conceptual Structures (ICCS'96), Sydney, Australia, Lecture Notes in Artificial Intelligence, LNCS 1115, Springer Verlag, p. 248-262, 1996.
[Sowa, 1984] J.F. Sowa. Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading, Massachusetts, 1984.
[Sowa, 1999] J.F. Sowa. Conceptual Graphs: DpANS. http://concept.cs.uah.edu/CG/Standard.html 1999.

12 Appendix

Appendix 1: RDF Schema of DRDFS

```
<rdf:RDF xml:lang='en'
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
xmlns:drdfs='http://www.inria.fr/acacia/drdfs-schema#' >

<rdfs:Class rdf:ID='DefinedClass'>
    <rdfs:subclassOf rdf: resource=
'http://www.w3.org/2000/01/rdf-schema#Class' />
</drdfs:DefinedClass>

<rdfs:Class rdf:ID='DefinedProperty'>
    <rdfs:subclassOf
resource='http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property' />
</drdfs:DefinedClass>

<rdfs:Class rdf:ID='Context'>
    <rdfs:subclassOf
resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdfs:Class>

<rdfs:Class rdf:ID='Axiom'>
    <rdfs:subclassOf resource='#Context' />
</rdfs:Class>

<rdfs:Class rdf:ID='Variable'>
    <rdfs:subclassOf
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdfs:Class>

<rdf:Property ID='hasDefinition'>
<rdfs:domain rdf:resource='#DefinedRelation' />
      <rdfs:domain rdf:resource='#DefinedConcept' />
    <rdfs:range rdf:resource='#Context' />
</rdf:Property>

<rdf:Property ID='referent'>
    <rdfs:domain
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
    <rdfs:range
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdf:Property>

<rdf:Property ID='isContextOf'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdf:Property>

<rdf:Property ID='parameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='formalParameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='firstFormalParameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='secondFormalParameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='if'>
    <rdfs:domain rdf:resource='#Axiom' />
    <rdfs:range rdf:resource='#Context' />
</rdf:Property>

<rdf:Property ID='if'>
```

```
    <rdfs:domain rdf:resource='#Axiom' />
    <rdfs:range rdf:resource='#Context' />
</rdf:Property>
</rdf:RDF>
```

Appendix 2: Definition of « WebPage » in DRDFS

```
<drdfs:DefinedClass rdf:ID='WebPage'>
    <rdfs:subclassOf rdf:resource='http://…#Document' />
    <drdfs:hasDefinition>
        <drdfs:Context>
            <drdfs:formalParameter rdf:resource='#x' />
            <drdfs:parameter rdf:resource='#y' />
            <drdfs:isContextOf>
                <Document>
                    <drdfs:referent rdf:resource='#x' />
                    <hasForRepresentationSystem>
                        <Format>
                        <drdfs:referent rdf:resource='#y' />
                        </Format>
                    </hasForRepresentationSystem>
                </Document>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#y'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
    </drdfs:hasDefinition>
</drdfs:DefinedClass>
```

Appendix 3: Definition of « colleague » in DRDFS

```
<drdfs:DefinedRelation rdf:ID='colleague' >
    <drdfs:hasDefinition>
        <drdfs:Context>
            <drdfs:formalParameter rdf:resource='#x' />
            <drdfs:formalParameter rdf:resource='#y' />
            <drdfs:parameter rdf:resource='#z' />
            <drdfs:isContextOf>
                <Person>
                    <drdfs:referent rdf:resource='#x'/>
                    <worksIn>
                        <Institute>
                            <drdfs:referent rdf:resource='#z'/>
                        </Institute>
                    </worksIn>
                </Person>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <Person>
                    <drdfs:referent rdf:resource='#y'/>
                    <worksIn>
                        <Institute
                            <drdfs:referent rdf:resource='#z'/>
                        </Institute>
                    </worksIn>
                </Person>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#z'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
    </drdfs:hasDefinition>
</drdfs:DefinedRelation>
```

Appendix 4: Representation of « If x is a colleague of y, then y is a colleague of x » in DRDFS

```
<drdfs:Axiom>
    <drdfs:formalParameter rdf:resource='#x' />
:   <drdfs:if>
        <drdfs:Context>
            <drdfs:isContextOf>
                <rdf:Description>
                <drdfs:referent rdf:resource='#x'/>
                <colleague>
                        <rdf:Description>
                            <drdfs:referent rdf:resource='#y'/>
                        </rdf:Description>
                    </colleague>
                </rdf:Description>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#y'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
:   </drdfs:if>
:   <drdfs:then>
        <drdfs:Context>
            <drdfs:isContextOf>
                <rdf:Description>
                <drdfs:referent rdf:resource='#y'/>
                <colleague>
                        <rdf:Description>
                            <drdfs:referent rdf:resource='#x'/>
                        </rdf:Description>
                    </colleague>
                </rdf:Description>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#x'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
    </drdfs:then>
</drdfs:Axiom>
```

Enterprise-standard ontology environments for intelligent e-business

Alan Flett
Alan.Flett@SemanticEdge.com

Mike Brown
Mike.Brown@SemanticEdge.com

SemanticEdge
www.SemanticEdge.com
Kaiserin Augusta Allee 10-11
Berlin, Germany

Abstract

This paper highlights SemanticEdge's use of ontologies within their much broader conversational e-commerce system. After sketching some of the problems in e-commerce and e-business, we introduce the conversational paradigm as applied to e-commerce. This paradigm requires the use of ontologies in many areas, and we go on to outline the major issues we face in applying ontologies, both from a technical and a methodological aspect. We then go on to outline more general issues facing ontologies which we believe will be crucial to ontology technology's acceptance within modern enterprise-standard information technology systems.

Keywords: Ontologies, e-business, e-commerce

1 Introduction

The aim of this paper is to provide a pragmatic perspective on the emerging information technology of ontologies; how it can help solve various information integration problems in electronic business, and how successfully it is being introduced into a leading-edge e-business company's business processes. How successfully this technology can be fostered from the research environment to becoming a useable commercial information technology is of the utmost importance for the development of network-based information access; that ontologies, in the form which we will discuss them here, have been studied in various esoteric fields within computer science, principally artificial intelligence, should alert the interested information technology professional to the potential weakness of such technology: namely lack of accepted standards, lack of methodological guidance and support, and lack of enterprise-standard environments and tools. These are, arguably, the crucial issues which govern the take up of any new technology. The fact that the technology works, or has a sound, well reasoned, argument for its existence and usage, is assumed as a given. Having been convinced of its usefulness, the interested information technology professional then starts asking the "mundane" questions which typically overlap little with research-oriented academia: How do I apply this to my business?; Are there existing standards?; Is there methodological support?; Is there a unified common accepted methodology?: Is the technology of a standard that I can trust in my enterprise information systems?; Are there people trained in these methodologies and technologies? Whilst the last issue is, admittedly, outside most people's control, especially in an emerging technology phase, the other issues are ones that should be addressed by parties interested in the successful fostering any new technology to maturity. These are the issues which we address in this paper. We seek to highlight those issues which we believe will affect the usability of ontology technology within enterprises (both at the technology level and business process level), and we make some suggestions as to what potential solutions might be. There are certainly more issues than those we highlight here which will affect the successful uptake of ontology technology (e.g., alternative technology, the current trials and tribulations of e-commerce business models), yet we believe those which we do expose are outstanding open issues for the field to address.

The outline of the papers is as follows. In Section Two, we outline some of the problems that e-business faces. In Section Three, we go on to introduce the conversational paradigm for e-business, and how ontologies help provide some of the functionality required by that it; ontologies are also introduced here. In Section Four we cover some problems of ontology technology and methodology we see as important, suggesting some solutions we believe may be beneficial. Section Six concludes our paper.

2 An introduction to e-business and its problems

People communicate imprecisely; computers take you literally. Language is rich and thoughts are multidimensional; computers have no room for variability or implied meanings. While this situation is

tolerated in a research environment, it is intolerable and even disastrous in commerce or in real life. For this reason, the socially adept computer has become the holy grail of e-commerce, and, in fact, of human-computer interactions in general. This kind of computer must be mindful of who it is talking to, remembering what was already stated and who said it. It must react appropriately within the context of a situation, remaining flexible and self-adjusting, and understanding of changes in intention or direction. It must be flexible in detecting and offering appropriate alternatives if a specific question can't be answered precisely. It must accept alternative forms of statements and be conversationally comfortable to talk to, offering multiple modes of input and output, with an easy conversational style. It must be skilled at negotiating, eliciting goals, offering alternatives in price, size, style, and be informative by offering suggestions to the user based on its understanding of available alternatives and the context of the conversation. Above all, it must be *socially adept*—a social detective capable of interpreting intentions, background, level of user-need from how queries are phrased. It must match a social model to the appropriate conversation script. It must be able to infer a user's level of frustration and respond appropriately and be able to query and respond appropriately within the context of a given situation.

That is an ambitious vision for e–commerce to realise. At present, there are various problems impeding the realisation of such a vision. These impediments break down into what may be called information integration problems and human-computer interaction problems. Issues from both categories will now be outlined.

Query rather than search

E-marketplaces are more confusing than their real world counterparts. Not only must products from several vendors be collected in the right place, but these products must be matched to the requirements or queries of the end user. Simple search systems can answer direct questions and match products to those questions when the terminology of both the product description and the query match. That is, most search techniques are keyword-based (including the most WWW search engines). This kind of search paradigm not only retrieves information which merely has the same terminology, but it also misses information which uses different terminology.

Static product search

However, search is just the beginning of a broader negotiation process in purchasing products (or, in the generic sense, of retrieving pertinent information). In a real marketplace, the buyers and the sellers come to an accommodation; buyers enter with a set of requirements, including price; so do vendors. These requirements are rarely met by any product; the price may be too high, the features of the product may be missing some of the initial requirements, or there may be no one product that proves a good match. At this point, both the buyer and the seller must decide what they are prepared to negotiate: Can the buyer pay a slightly higher price?; Could the vendor add an additional feature affordably?

Unstructured text

In addition, most e-marketplace technologies rely on structured databases to store and retrieve data. However, structured databases do not have the capability to handle the unstructured, unpredictable and complex documents that are typical of both product descriptions and other related information such as product reviews. The issues then become those of extracting the salient information from such unstructured text sources, or of structuring such documents with meaningful, machine-processable, annotation—all complex problems.

Heterogeneously modelled data

In the web environment, users are further confounded by their lack of knowledge about how any system is structured. Interactions must be forgiving—capable of handling any form of query and responding with good matches to vague questions. Whilst most of today's e-marketplaces stop at search, some of the more advanced also categorise their product offerings so that roughly similar products are retrieved together. While categorisation improves the chances that the right product will be retrieved, it is seldom an easy process to automate. In reality, most categorised systems are a cobbled together collection of automatic features and manual labour. Semantically mapping between and among catalogues is a complex process; each vendor describes his wares in different terms. Equivalent products must be determined, and judging equivalence is usually beyond most automatic systems. Also, manual systems are not scalable; it is not possible to keep constantly changing systems up to date in a timely manner using manual labour.

Cultural and individual personalisation

Today's increasingly global market, especially as it is manifested in the Web, demands support for multilingual communication, information access and transactions. It is expected that the demographics of the Web will dramatically shift away from the US and away from English over the next few years. An e-commerce application that can only converse in English, is of no use to a native Spanish speaker. Multilingual support can be important to employees and business partners as well as potential customers. So, too, for someone searching a catalogue for a

product or service to purchase. But language is only the "tip of the iceberg"; these systems must also be localised to the modes of data presentation (e.g., currency, dimensions), product and business regulations, business practices and cultural norms of the user's own country.

Telecoms infrastructure media

E-marketplaces also need to meet the needs of an increasingly mobile user population. The mobile user wants to purchase products on the run. This means that a wireless device such as a PDA or WAP telephone is the interaction venue of choice. Large text files or graphics are useless in this environment. Voice input and output would greatly enhance such mobile and wireless transactions. Speech recognition and speech generation are, however, only functionally useful in very limited ways. One of these limitations is the lack of strategic interactive knowledge, that is, the ability to hold dialogs with users interested in fulfilling some goal, be it access to information or to exercise a transaction for some desired product.

All these problems with current e-commerce/-business require solutions. One of the most talked about solutions to many of the information search and integration issues is ontologies. Ontologies promise a lot; a shared and common understanding of some domain between application systems, and between them and people. In terms of human-computer integration, conversational systems are also being talked about as the ideal way automatic e-business should be executed. It is to these conversational systems, and how ontologies will aid in providing these systems with various functionalities, which we turn our attention to next.

3 An introduction to an e-business solution: ontologies meet conversational systems

3.1 The requirements

All business transactions consist of a complex set of interactions among vendors, buyers, and the information that exists surrounding the product or service—evaluations of products, notions of reliability, stylishness, appropriate price, service, etc. The marketplace is heavily dependent on such information, and is, in fact, a specialised information system. It matches vendors' offerings with buyers' needs. It informs the consumer so that he understands his options. This process is ideally carried out through a dialogue that helps both sides adjust their offerings and requirements. That is, the technology has to be an adaptive system based on natural language understanding. It has to be able to discern the meaning and even the underlying intent of a question. It also has to adapt during the course of a negotiation with a customer, as the customer's requirements narrow or change in reaction to the availability or suitability of products in the marketplace. It has to have multilingual input and output as well as the capability for speech recognition and generation to make it appealing in an increasingly global marketplace.

Furthermore, any such system must be integrated and robust enough to sit on top of the complex functions that constitute modern information-mediated transactions, that is: context-dependent search; multilingual; product retrieval across multiple suppliers, each with different descriptions of analogous products; continuous updating of products, prices and other information; tracking of orders; feedback to suppliers on successful/unsuccessful products and the products that are requested but are unavailable; fast response time: fast updating; sticky features to keep people on the site; and an improvement on current systems, both in quality of service and in cost savings.

3.2 The solution

SemanticEdge has developed a state of the art multilingual natural language (text and voice) dialog system capable of handling dialogs with humans wanting to access information, for example, to purchase products and services. The technology extends naturally to Customer Relations Management (CRM) and other e-business functions. This technology depends on several distinct technology areas within Artificial Intelligence: natural language processing, including deep language processing and statistical analyses; machine learning, including inductive learning; speech recognition; automated dialog generation, both user and content specific; and knowledge representation and ontologies.

The system mediates between humans and information. That is, it mediates between an information space and a human's conceptualisation of that information space; for example, between a product space and a customer's conceptualisation of that product space, and how they will consequently go about searching and querying that product space. Users hold negotiations with the system, which is mediating access to the product spaces, and it will ask questions of them. This requires the system to have the ability to guide those dialogs according to a representation of that product space. This ability to a large extent is supported by ontologies. Not only does the technology model products objectively, as might be done with a sophisticated database system, but we also model subjective quality judgements that consumers tend to use when conceptualising the product space before them. These subjective, ad hoc, categories gives the system the ability to communicate

to the consumer in a human friendly way, in a way that is, in terms of the ontological commitments made by the system, similar to those of the typical customer or user. These human-oriented aspects are further enhanced by other technologies within the system, such as user models of consumer reaction to the dialog process as it happens.

3.3 Ontologies' role in the solution

As noted, many of these information search, integration, and modelling, functionalities are supported by ontologies. SemanticEdge uses both cutting-edge ontology editors and environments from third parties and its own ontology technology to build both domain and user ontologies. By domain ontology we mean both objective and subjective ontologies. Objective ontologies model the standard product descriptions typically released by product manufactures. These include attributes such as weight, price, product features, and so on. By subjective ontologies we mean those attributes of a product which are somehow generated by consumers' (common) conceptualisation of the product, such as a "fast", "noisy", "family" printer, and so on. These subjective quality attributes and ad hoc categories are typically the main communication vocabulary that people use to conceptualise the product/information space and consequently the terminology they use in dialogs. Also, user ontologies play the role of modelling different types of user and mapping those different consumers, and their attendant different requirements, to different dialog strategies and ultimately to different products and services. Since these knowledge bases are concept based, they are language independent—an important feature in a multilingual system that must retrieve in any language; new languages need only to be mapped to concepts, not translated term by term.

This ontology building effort is large in scale and complexity and its management is non-trivial. It requires extensive automation and support from intelligent tools. It is our experience that at present, whilst competent inference engines and editors are available from third parties, large parts of what might be called a comprehensive ontology engineering workbench or environment are missing; worse, it is those parts which would allow for the efficient application of ontology technology in enterprises which are missing. Consequently, SemanticEdge has developed its own set of ontology building technologies. These technologies allow for the capture of large amounts of instance level information from unstructured through to highly structured sources; that is, to populate an already existing intentional structure. SemanticEdge is also developing ontology learning technology which will help in the burden of building the intentional structure; that is, acquiring the concepts and conceptual relations (e.g., subclass relations) from free text as well as from more structured sources. The technology being developed uses a mixture of adaptive pattern recognition, inductive learning, several machine learning technologies, and an extensive set of heuristics. SemanticEdge is also currently involved in several academic projects, such as OntoWeb [4], and industry consortia all with the common goal of working to develop more advanced shared ontology technologies, to allow for their efficient and efficacious application in enterprises.

What are ontologies anyway?

The standard working definition to which most ontologists refer to is that of Gruber [1]. Others have defined ontologies in largely similar ways [2]. The common characteristics an ontology is supposed to have are that it should be a *formal, explicit specification of a shared conceptualisation*. That is, ontologies (in the sense propagated here, and the one recognised in artificial intelligence) should be a conceptualisation of some phenomena. How complex this conceptualisation is up to the conceptualisor, and with regard to the formal specification, also depends on how expressive the specification language is (humans having very expressive specification languages being able to have very complex ontologies of their world). Formality means that that we may be able to automatically map and reason with our specification, typically with the aim of having machines reason with the various ontological knowledge. The notion of the ontology being shared means that the specifications made are to some extent common throughout some group of members. This conceptualisation may have been arrived at by common consent or not, but the members of the group are said to have *committed* themselves to the ontology.

The elements of ontologies

Ontology languages are the formalising structure which represent the domain/universe of discourse or world we are interested in. To accomplish this, editors are typically used which bypass many of the textual characteristics of specification languages and provide more perspicuous and efficient means for developing, maintaining, and modifying ontologies. Taking the ontology editor metaphor to another higher level, we end up with an ontology environment metaphor, where various potentially time saving and quality increasing features are to be had (e.g., modularisation, versioning, and reuse mechanisms in general, verification and validation). Once the ontology has been developed, we can think of performing reasoning with the ontology, requiring inference engines; which may also be part of the development environment, thereby supporting intelligent editing, and perhaps enabling debugging and tuning of the ontology from

reasoning efficiency and competency viewpoints. The interested reader might want to read [2] for a detailed exposition of ontology technology.

This brief overview of how ontology technology fits into SemanticEdge's conversational system motivates some important issues in the application of ontology technology and methodology. One of the most important is that it requires the ability to manage large ontologies. This scalability requirement motivates several other ontology engineering issues, including: acquisition; visualisation; modularization and versioning; reasoning transparency; multitasking; competency; and methodology. In the remainder of this paper, we will explore all of these issues more fully.

4 Issues in enterprise-standard ontology environments

Many of the above elements are now discussed, but some are not. Notably, languages are not discussed, as these are, arguably, where academia can be and has been of greatest input; these languages are complex, the semantics issues complex, and not many people outside academia are competent to go around designing semantically well designed ontology languages. The other elements are largely those of engineering application; they require manpower, capital, and management.

4.1 Acquisition

The acquisition of ontological knowledge and the instantiation of such ontologies is a prime issue in the building of large ontologies. There are several distinct knowledge acquisition phases in ontology building. There is the initial acquisition phase, where the structure and terms of the domain are acquired and represented; this is also part of the conceptualisation phase mentioned before. There is then the acquisition of rules and axioms of the domain; this is also part of conceptualisation. Lastly there is the acquisition of instances, the instantiation of the ontology with facts to be reasoned with. The aim in all these phases of acquisition should be to acquire what needs to be acquired quickly, efficiently, correctly, and with as little effort as possible.

The different phases impose different requirements, and the goal of making them automatic are more realisable in certain phases than in others. The least amenable to our wishes are the first and second, that of acquiring conceptual knowledge of the terms, structure, rules and axioms, of the domain; that is, the conceptualisation of a domain is a skilled process. It is in the last phase, the acquisition of instances, that most can be accomplished, though this a non-trivial task when the source data is informal, such as text in web pages.

The automated acquisition of such instances is something to which SemanticEdge has invested considerable resources in. Proprietary information extraction technology has been developed to support the acquisition of information from unstructured to more structured sources. A screenshot of the GUI can be seen in Figure 1. SemanticEdge is among a growing number of companies that offer specialised technology for carrying out this information extraction task. A number of trainable and self-learning Artificial Intelligence (AI) technologies are encapsulated inside a single Information Extraction Engine. These AI technologies can be configured to map any number of different product catalogue formats onto a single intermediate, predefined product schema. From this schema, information can be exported into one or more formalised representations (including ontology languages).

Export involves two basic steps:

- **Normalisation**: This can simply involve mapping one of a number of synonyms for a given piece of product information onto a single predefined symbol. It can also involve more complex normalisation rules such as converting numeric attribute values that can be given in one of a number of units onto a single standard unit.

- **Generation of Export Syntax**: Through the attachment of formatting rules to the intermediate product schema, high flexibility in the export format can be achieved, and as noted, the information can be output to ontology languages, such as, for example, F-logic.

One further requirement is that the acquisition technology should be useable by non-knowledge engineers. This again is a distinguishing feature between the three phases, with again, the first two being more knowledge engineer heavy, whereas the third, if done properly, can be accomplished by non-experts in conceptual modelling. Of course, there is more to acquisition, especially when considering the first two phases where the process is more one conceptualisation. Here the progressing conceptualisation and formalisation would be partly helped by perspicuous visualisation, and it is that which we will now look at in more detail.

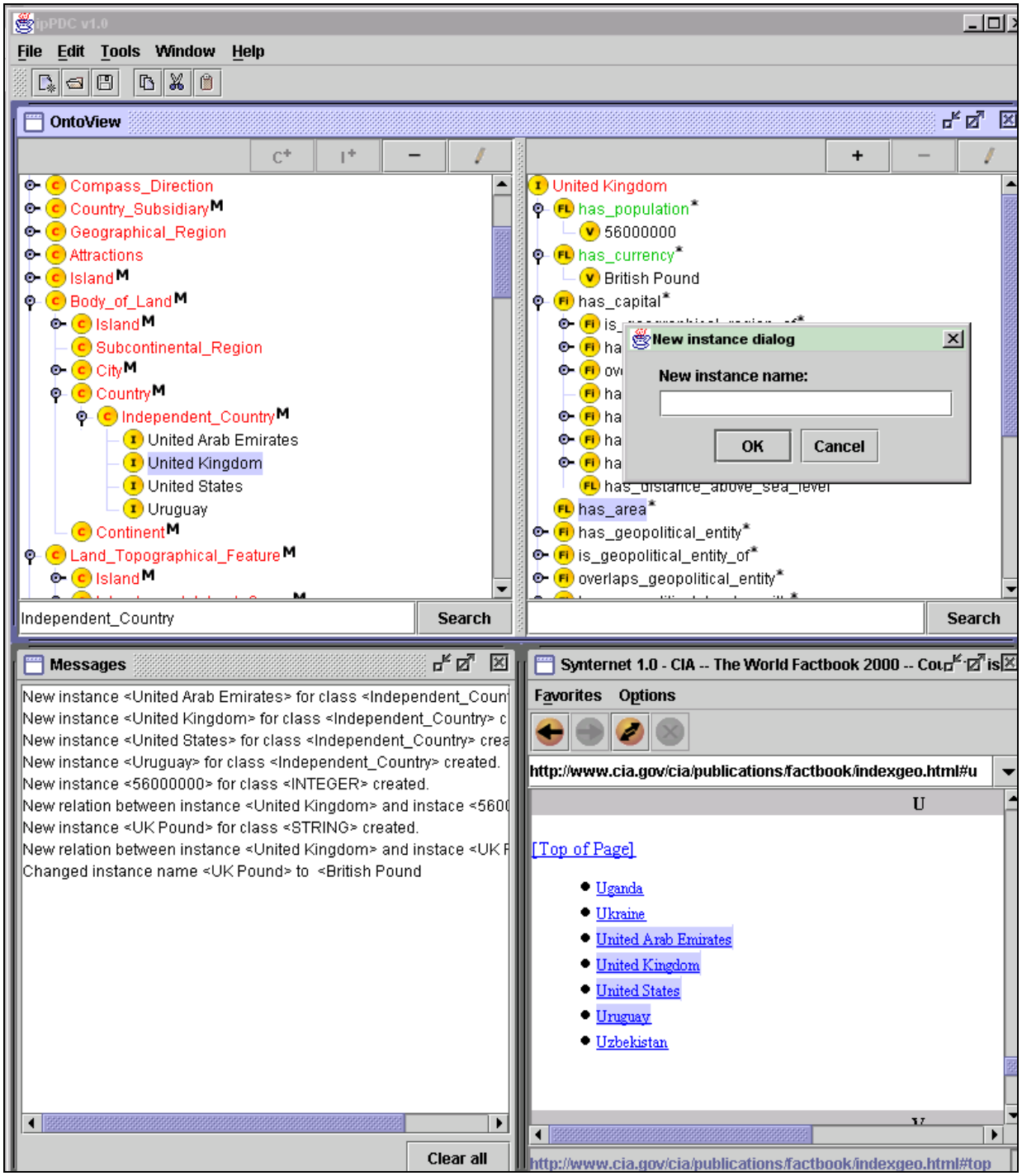


Figure 1: SemanticEdge has developed the sePDC to enable the acquisition of product instances. This extensional information has to conform to the imported ontology, and a number of ontology formats, including F-logic, can be accommodated. Here we are capturing some new instances of the Country concept from the CIA World Fact-book.

4.2 Visualisation

Visualisation has dogged conceptual modelling for years, with the taxonomic metaphor often blinding all to any other conceptualisation of how ontological knowledge might be presented. While a high quality implementation of a taxonomic presentation is both

important and useful, it is not the whole story. Certainly, for acquiring taxonomic subclass relations, it is probably a fine paradigm to use. The issue become more complex when one considers that the visualisation should somehow convey other, more complex, aspects of the conceptualisation. This visualisation of a conceptualisation has to include not only the terms but also the axioms in relation to the terms they relate to, and so on; although the visualisation of axioms themselves is a tricky issue. Further, editors which somehow support the user in defining axioms must be welcomed.

Visualisation should aid in the conceptualisation of the domain, as well as being some kind of first-pass validation of the ontology's commitments.

4.3 Modularization and versioning

One issue which we have noted as being of importance to the ontology engineering issue is a very important lesson learnt in software engineering, that is, of reuse and modularization. The benefits of reuse and modularization have been often repeated, and we will do so here again in the light of ontologies.

Reuse is a good thing. It reduces effort through not requiring basic (or, nowadays, more complex) components to be built from scratch, increases quality through the reuse of quality components, and thereby reduces effort and costs. But reuse also brings problems of its own making such as building the most reusable components possible and finding and integrating the most up to date versions of these components into a working and consistent whole. In the ontology world, the merging, alignment, version control, and so on, of ontologies, is only now beginning to make it into tools, and has a long way to go. These are, however, going to be crucial issues which will become very important in years to come, as more and more ontologies are built and available on the WWW, where different components might be inconsistent with each other, and out of date versions abound. We can imagine a time when we will be able to select from a library of well designed, up to date, mutually consistent components which can be easily, even automatically, integrated, all done in some kind of developer studio environment.

4.4 Reasoning transparency

The are several issues regarding reasoning which require close attention. Reasoning in the small and in the large are totally different propositions. Issues such as inference profiling, debugging, inference efficiency concerns, how the modelling affects the reasoning to be performed, what the competency of the ontology implies for the completeness of the reasoning, logically erroneous axioms causing problems (circular axioms), and so on, all require attention, especially when the ontology is scaled up. For instance, let us assume we have a slow query. Is this slowness an implementation problem such as a bug in the reasoner, is it a problem of having too complex a model, is the model okay but the reasoner is providing us with too much reasoning, for which we have no use as regards the competency we have decided the ontology should support, is the reasoner sound and complete but not up to the job?, or is it simply a slow machine on which we are running our system? To answer these questions requires the ability to observe the reasoner in action and to have access to some kind of statistics and summary information. This aspect of ontologies is crucial, as the model-axiom-reasoner interaction is a non-trivial one and the space of possible interactions is immense and totally unpredictable in terms of the efficiency of the concerted artefact. This is something, that, for instance, databases do not suffer from to anything like the same extent. Indeed, this issue goes to the heart of much knowledge representation research in that it encompasses completeness, soundness, tractability, and so on.

It is our opinion that reasoners cannot be black boxes in to which no one may look. For the engineering of large ontologies, it is necessary to be able to at least appreciate the problems and to try and solve them however we can or is allowed by the reasoner paradigm in question. Altering the completeness and soundness characteristics is probably an extreme way of solving any potential problem, but at the very least, one should be aware of what inferences are being computed and why, so that, for example, over burdensome axioms may be modified if the competency specification allows it. These issues of reasoning efficiency become even more important when considering that these reasoners may very well be on-line knowledge servers to which multiple users (e.g., hundreds) may be accessing. It is to the issue of multitasking and multiuser access that we turn to next.

4.5 Multitasking

If ontologies, the models and reasoners, are to become trusted on-line servers of knowledge, then they must have certain features developed over the years in the database community. For example, the capability to handle multiuser access would seem to be the minimum. Multiuser access may be querying, browsing, or editing. There is much demand for all at SemanticEdge, where ontological tools and the conceptual models they support are required to be browsed by several people, who have to keep their work somehow consistent with the ontological commitments already made. And, indeed, many people have concepts which they might want to add, if suitably qualified to do so. And there are other issues, such as stability and reliability. This implies some

professional engineering support and development to take ontologies to the next level of enterprise integration.

4.6 Competency

The notion of competency is one of the most important issues in conceptual modelling. It is important in that when we model a domain, we expect the resultant concerted artefact of model, axioms and inference engine to be competent with regards to the queries we should wish to ask of it. It is in the interaction of the these three components of any ontology that the complexity arises, and we require someway of assessing whether or not we have been successful in building the ontology artefact which satisfies our requirements. For example large amounts of global terminological assertions such as partitions and so on can have quite large effects on the kind of deductions possible (as well as reasoning efficiency, see earlier section on reasoning transparency).

Deciding when an ontology should be declared competent is not easy. One solution is to have a test harness where several representative and/or complex queries can be entered and run on the ontology. If these cases give satisfactory responses then one might conclude that the ontology is competent and leave it at that. Alternatively, one could have a comprehensive set of queries somehow automatically generated which comprehensively exercises the ontology's competency (one may even want such functionality so as to cache the results of our ontology if we decide that its performance is not good enough to have as part of an online, live, system). The returned answers would then have to be somehow assessed and a judgement made on the ontology's competency.

4.7 Methodology and Ontology

One of the most challenging aspects of ontology work is developing new ontology structures, capable of representing what is intended. This is typically not easy when one wanders, even slightly, from the well-trodden path of EER modelling commonly practised in the database (and OO software engineering) world. Representing other kinds of world phenomena, well axiomatised, is not a trivial task. Upper level ontologies, where a solid, well thought out, conceptual structure, offers the benefits of providing methodological support for modelling and Ontology. For example, part-whole knowledge is one of the most common ontological structures humans use to think of the world, and it is also one of the more complex representation and reasoning paradigms to get right. Methodological support in this area, as well as the definition of various well-conceptualised upper-level concepts, is crucial for the development of ontologies that will talk to each other. Without such support, it is quite possible that ontologies unable to be integrated will find their way into various resources, and significantly harm the information integration dream that many believe ontologies offer. After all, the word Ontology, as per our working definition given earlier, apparently for many people has a notion of a common and shared meaning of terms, and it therefore seems fairly reasonable to hope for at least a common methodological upper level Ontology which people can use and extend.

5 Conclusion

E-marketplaces accentuate several dimensions of the product buying process. They bring incomprehensible scale, and thus the consequent problem of enabling a customer to, indeed, comprehend and interact with this space. This large space brings problems of heterogeneity (how to find all similar products), search (how to find your products amongst the hundreds or thousands available), choice (is there too much choice now available for the average customer), and optimality (is the customers choice the optimal one in the space of his product options). These problems of scale are not the only problems in allowing human customers to interface optimally with their chosen product space; there are conceptual problems independent of scale. These problems lie in allowing a human customer to conceptualise and communicate in as natural a way as possible with the buying process. This consequently means depending on complex natural-language and knowledge-based systems (including ontologies) supporting a dialog or negotiation. SemanticEdge is currently developing such cutting-edge technology (including ontology technology) which will allow it to so facilitate this natural transactional process between consumers and the products they wish to buy.

However, it is our opinion that ontology engineering has a long way to go before becoming truly enterprise-standard. It must embrace many of the engineering paradigms of object-oriented (OO) software engineering and database engineering that have become de facto over the last few years. Owing to the added complexity of ontologies in that they have inference engines, knowledge models, axioms and rules, all interacting in a non-trivial way, there are extra problems and complexity to be managed. At present, this additional complexity is not managed at all and is largely hidden from the user. Only symptoms such as slow queries, strange deductions, hanging inference engines and incompetent answers surface to alert the ontology engineer to problems. For the successful engineering of ontologies, it is essential that this complexity be better managed and thoughtfully exposed, so that when problems do occur there are recourses to take and information to be examined. This along with the substantial beefing-up of ontology environments will allow them to become

the next essential components in future distributed ontology-enabled information systems (e.g., resources on the WWW). Such information environments are beginning to be seriously discussed by many as the next stage in the evolution of, for example, the WWW—the Semantic Web—with such initiatives as DAML [3], where ontology languages such as DAML+OIL are being designed to support this, reinforcing this belief.

6 Acknowledgements

We acknowledge that part of this paper was based on the SemanticEdge white paper as written by Susan Feldman of IDC.

7 References

[1] T. R. Gruber: A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 5:199—220, 1993.

[2] D. Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag, Berlin, 2001.

[3] DAML: http://www.daml.org

[4] OntoWeb: http://www.ontoweb.org

Building Business Applications By Integrating Heterogeneous Repositories Based on Ontologies

Noriaki Izumi and **Takahira Yamaguchi**
Dept. Computer Science, Shizuoka University
3–5–1 Johoku Hamamatsu Shizuoka 432-8011 JAPAN
{izumi, yamaguti}@shizuoka.ac.jp

Abstract

This report proposes an integrated support methodology for constructing business models including employing new business models, transplanting existing business activities to computers, and decision making support in employing new environment of computers. In order to model enterprises and business activities and to implement them as software applications, heterogeneous repositories in different granularities of business models are integrated based on ontologies. By devising a framework, which picks the main concepts of repositories up and make correspondence among them, our framework achieves the unified reuse of existing repositories of business activities and software libraries. We have implemented the prototype system by JAVA and confirmed that it supports us in various phases of business application development including business model manifestation, detailed business model definition and an implementation of business software applications.

1 Introduction

Due to the rapid change of business environments introducing the Internet, it becomes very important to achieve the rapid adaptation of the corporate structure by employing a variety of heterogineus reusable componets such as business models, best practices, software libraries, and so on.

Because of the above context, a lot of research and development projects, which construct business repositories of various concepts and ideas relating to business activities, have been activated.

In the field of MS (Management Science), one of the famous results is the e-business Process Handbook project[MIT Process Handbook Project] carried out by MIT. The e-business Process Handbook is a substantial contribution as a business repository, which contains approximately 4,600 definitions of business activities from abstract processes to the specialized one to the business over the Internet. Its formality, however, is not strict since the most part of the definitions are described with natural language.

From the viewpoint of the formality on the process specification, there is the enterprise ontology[Uschold et.al. 1998] of Edinburgh University in the field of artificial intelligence. Its formality is very strong and it contributes the reuse of business models nevertheless it covers only so general and abstract concepts that it is very hard to construct concrete business models with operability.

On the other hand, one of the developed library for building knowledge systems on the concrete level is the inference catalogue of Common KADS methodology[Schreiber et.al. 1999; Common KADS]. Common KADS is utilized for analysis and development of knowledge systems and offers the language and primitives to clarify conceptual models. In late years, special method library REPOSIT is proposed for the implementation of inference primitives of Common KADS, but the methodology of the application construction including requirement analysis and development is still examined.

Owing to the difference of purposes among those repositories, when we try to build the real applications based on the existing domain, the integrated support are hardly performed in the whole process of the construction and the reengineering of business applications.

So, in order to achieve the unified support of the development, the computing framework, which integrates the different sorts of repositories, supports dynamic, should be developed for the construction of business models and applications.

From above-mentioned background, we propose the development methodology of business applications based on ontologies with reusable repositories such as e-business process handbook, Common KADS and REPOSIT.

In order to construct the business models and to implement them based on the existing domain, we rebuild the heterogeneous repositories into two repositories on different-grain-levels: the business specification repository on the level of business activities and the business software repository on the level of software applications.

It is the main characteristic of this work that our proposed framework enables integrated support of modeling business activities from business documents, and constructing business application from business models, by developing a platform to create the relationships dynamically between the descriptions of the business specification repository and the business software repository. We have implemented the prototype system by JAVA and confirmed that it supports us in various phases of business application development including

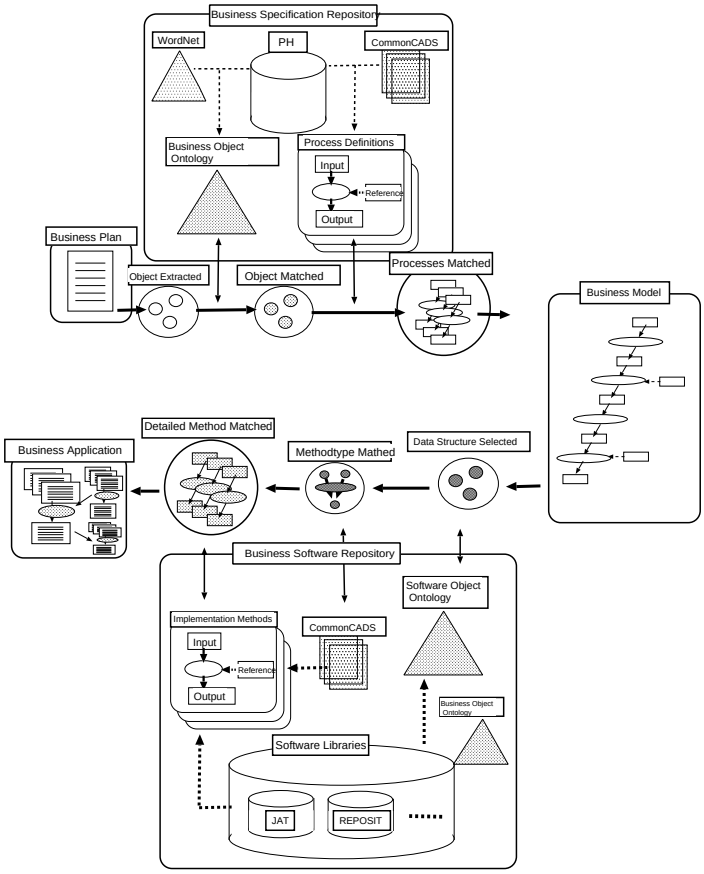


Figure 1: Overview of Development Support

business model manifestation, detailed business model definition and an implementation of business software applications. Since our work can be applied to advanced management judgment on introducing new business models, re-engineering of existing business processes, and employing new environment for business computing, we expect the repercussion effect on the management activities as the result of the information integration.

2 Overview of Proposed Development Support

In order to achieve the unified treatment of models on different levels such as making business models clear, implementing the detailed models, building software applications and so on, our research aims at the establishment of integrated support from the analysis level to the implementation level. Our standpoint of the integration and the reuse of heterogeneous repositories is to obtain the key structure of each repositories, which corresponds to noun concepts, and to relate them each other to bridge the whole structure of processes and activities including verb concepts.

The key idea to achieve our aim is how to extract the verb and noun concepts and their relationship as the common structure of information from the heterogeneous repositories. The target of development support is the construction of business application as a business models obtained from business documents, which contains the facility such as communication with users through network via E-mail or Web, and the file system sharing with the users for customer relationships and data management. Brief description of business application development through the repositories integration by our framework is as shown in Figure 1.

At first, in order to pick the key concepts up from business documents, we construct the business specification repository including ontologies of noun and verb concepts. Noun concepts are extracted from the e-business Process Handbook and classified into the business object ontology according to WordNet[Fellbaum 1998] as the general concept ontology provided by Princeton University. Verb concepts of e-business Process Handbook are classified by using co-occurrence information of noun concepts. By making reference to the history database of the correspondence between words of documents and ontologies of nouns and verbs, a business model is constructed by the business specification repository.

Next, as a repository for transplantation of the business model to a business software application, we provide the business software repository including method libraries. In order to correspond activities of business models to software mod-

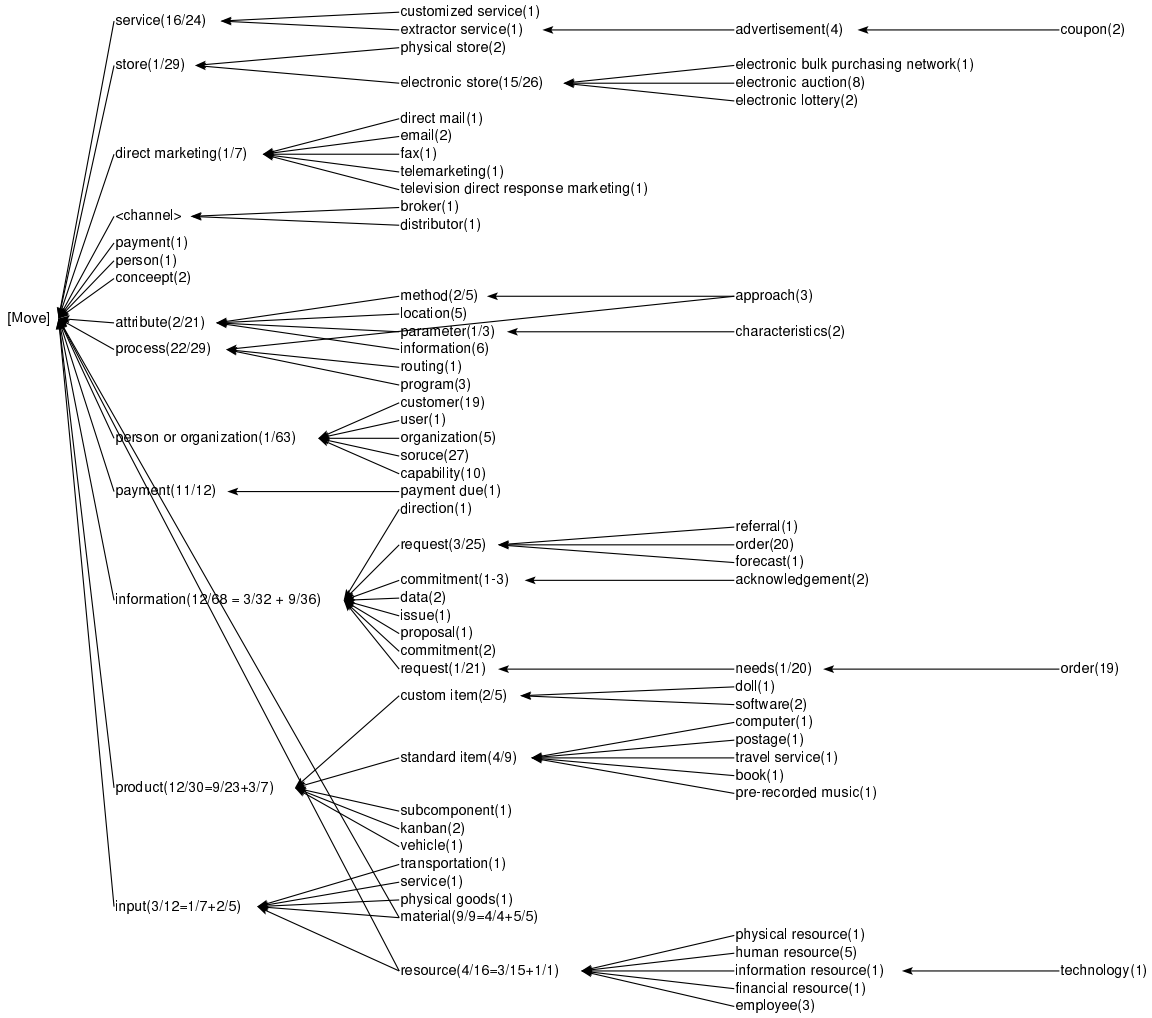


Figure 2: Object Tree Structure Obtained (a part of it)

ules, we construct a library of typical patterns of the input-output relations and the module structure of software systems based on JAT (JAVA Agent Template)[Petrie 1996] provided by Stanford University. Each pattern is formalized as a combination of inference primitives of Common KADS by introducing a software object ontology that provides the classification of objects with control and data structures. By consulting the software object ontology and REPOSIT[Izumi et.al. 1999b] provided by Shizuoka University as a library of the implementation patterns, the business model, obtained on the previous stage, are supplemented with control structures of software codes. According to the frequency and history of corresponding among three libraries of JAT, Common KADS and REPOSIT, a detailed model of the software application are obtained based on the software object ontology.

3 Construction of Business Models

In this work, we employ the e-business Process Handbook of MIT, called Process Handbook for short, as a library classifying business activities. In order to construct the business specification repository by extracting the required information, WordNet is also employed as a general lexical repository. First, the business specification repository is provided as a key structure bridging the business documents and Process Handbook. Then, the wrapper framework is constructed as an extract method of required activities from a repository of Process Handbook.

3.1 Building Business Object Ontology

When the abstraction degree of a business plan is high, a verb concept of an activity in the business plan is often vague for specification makers due to the difference of viewpoints on the definitions. In contrast, a noun concept of the activity is comparatively clear and appears regularly in the document. In order to classify the noun concepts extracted from Process Handbook, we employ the WordNet as a general ontology that contains over 17,000 concepts. However, if we utilize WordNet as it is, the number of candidate explodes because of the variety of the word's meaning and the ambiguity of the word in the document.

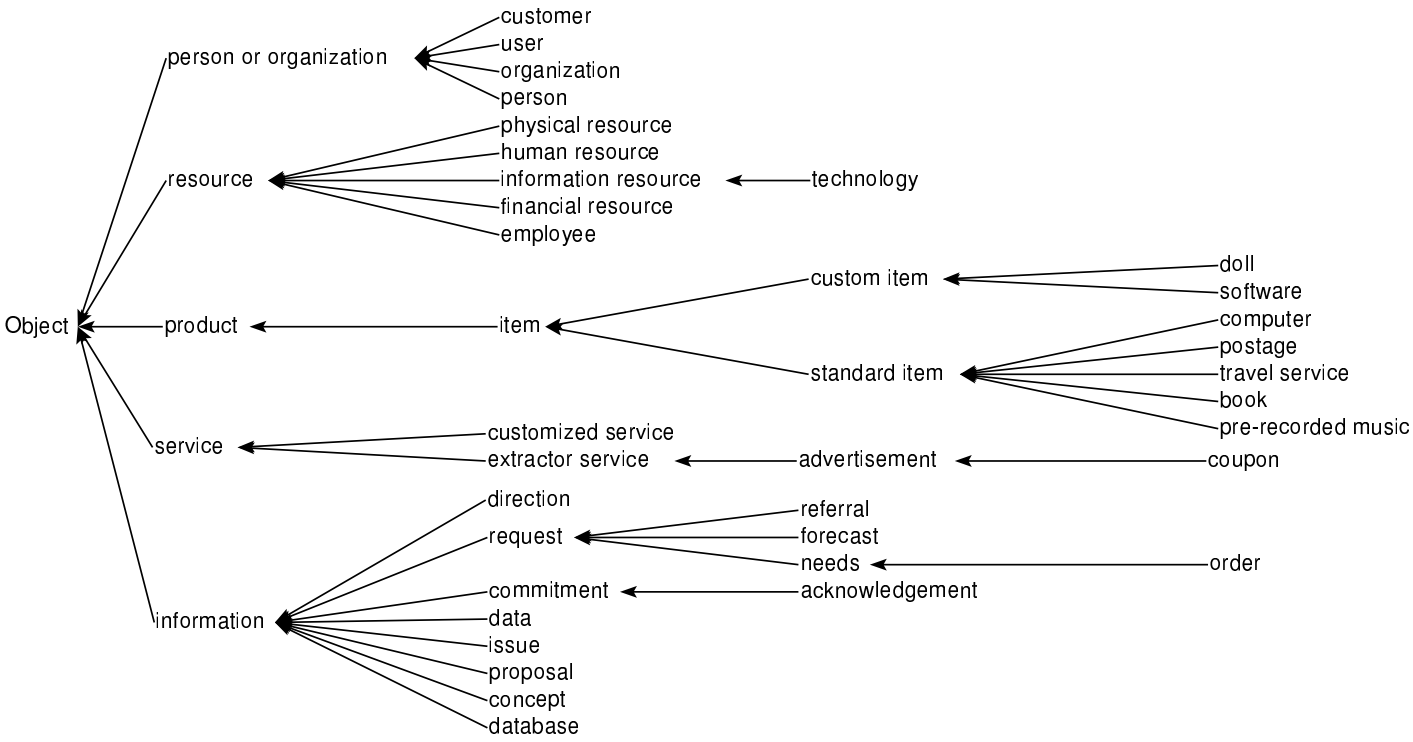


Figure 3: Business Object Ontology (a part of it)

So, in order to classify the noun concepts appearing in Process Handbook, we choose the major concepts with respect to the degree of abstraction and frequency by using WordNet in the following way.

First, noun concepts consisting Process Handbook are extracted and rearranged into a similar structure to the activities of Process Handbook. Then, the frequency of each word is counted and sum up with respect to the inherited structure. A structured tree of the words with the attribute of appearance frequency is obtained as shown in Figure 2. In the figure, the number of appearance is given in the fractional expression where the numerator corresponds the frequency of appearance as it is and the denominator means the inherited frequency of sub-concepts. The addition expression is a trace, which indicated the word, appears in the different position of the tree structure.

As the criteria to select a major noun concept, we pick the concepts with the more frequency up and rebuilt into an upper ontology. According to the WordNet's structure, we repeat the same way described above and define the substructure of concepts obtained above as an upper ontology, provided that the priority of the meaning is given to business domain over the relation of WordNet. Furthermore, when we fix the top ontology for constructing business domain ontology, concept drift, that is a kind of semantic shift on a specific domain, often occurs and causes inefficiency on building ontologies. Due to reduce the cost of construction, we employ the methodology for resolving the concept drift[Yamaguchi 1999]. Figure 3 shows the structure of the business object ontology obtained.

3.2 Determining Business Activities

When obtaining the definition of the business activity corresponding to a business document, it is difficult to utilize the hierarchical structure of the process handbook because of the gap between words of actual documents and process handbook. On this account, we construct the business object ontology, as a top ontology, which bridges the variety of words in documents and nouns in Process Handbook. In order to identify business activities from a sentence given by a user, we devise an extraction mechanism as a wrapper for Process Handbook based on the business object ontology.

The wrapper tool is composed of the databases of the following information. First, the co-occurrence information of noun concepts in the definition of a business activity is obtained and classified with respect to the structure of the business object ontology. Then, the information is made accessible as the database of the co-occurrence. At the same time, the frequency information is also available in collecting the co-occurrence one. By using both of the co-occurrence and frequency information, the wrapper tool help us to search the definition of activities in the space of Process Handbook. The proto-typing tool is shown in Figure 3.

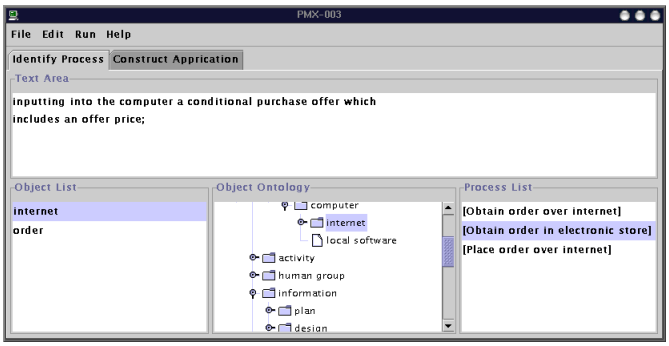


Figure 4: The Wrapper Tool

4 Building Business Applications

4.1 Building Software Object Ontology for Reuse of Libraries

In the same way of the business object ontology, the software object ontology (Figure 5) is constructed as a domain independent ontology from the libraries intended to employ. The software object ontology gives words for expanding domain ontologies such as building a set, picking up an atom of a set, indicating a calculation stage, data structures for implementation details and so on.

On the purpose of the developing business applications from the business model obtained above, a detailed definition of each activity of the model is required. In order to give the activities the operational information that is used for application development, we prepare the library of the application template, which defines the structure of the part of application in the fashion of the knowledge system development with respect to the software object ontology. By constructing the business software repository with the reusable template of REPOSIT, Common KADS, JAT and historical databases, the business application is obtained.

4.2 Model Refinement Based on Application Templates

We consider the structure of business applications based on the agent architecture to be composed by the inference engine to attain a task, the sensor to get the information of the outside and the effector to carry out their task. The sensor is characterized by the following three functions:

(1) the function that accesses the inside and the outside resources of the agent,

(2) the function that examines the place and the contents of resources,

(3) a function to acquire a message from the user and to interpret the message.

The effector is defined by two of the next:

(a) the function to form and modify the inside and the outside resources of the agent,

(b) a function to make and to send a message to the user.

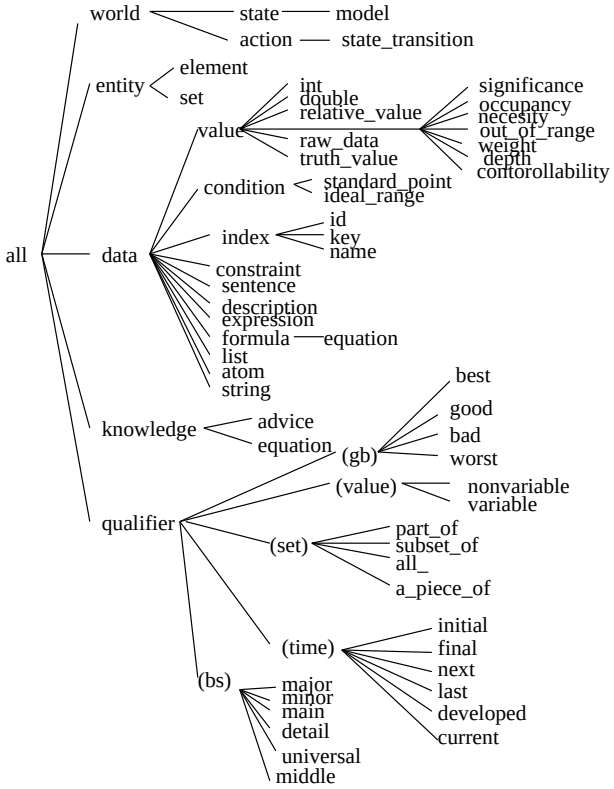


Figure 5: Standard data hierarchy

The framework of the combination with the above functions and the inference engine is organized as agent templates by referring to JAT (JAVA Agent Template) of the Stanford University. Furthermore, detailed templates, corresponding to eight types of communication models given by Common KADS, are formed as interaction templates with the user and resources (Figure 6).

4.3 Building Reusable Templates for Implementation of Applications

From the importance of a unified language for the reflection of the change on a business model, we rebuild and extend inference primitives of Common KADS into “REPOSIT (REusable Pieces Of Specification-Implementation Templates)” which combines declarative semantics employed in Common KADS and procedural semantics like Prolog. A unit of a description in REPOSIT, defined as a relationship among input, output and reference knowledge, is called a “unit function”. A set of unit functions is rebuilt into the method ontology by abstracting knowledge types of input, output and reference.

Furthermore, patterns of a combination of unit functions, which appear frequently in the development process, are gathered, sorted out and constructed as a method library based on the following standpoint:

(1) providing refinement policies,

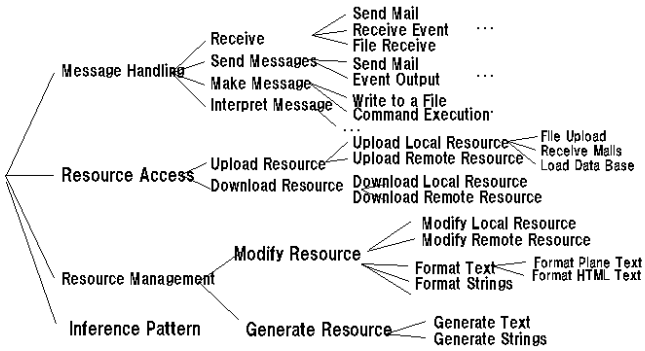


Figure 6: Primitives of Application Templates

(2) standardizing a way of the knowledge (data) management,

(3) classifying the adding patterns of control structures given to specifications.

In order to keep a correspondence between descriptions of specifications and implementations, REPOSIT supports step-by-step operationalization of an abstract description into a detailed implementation model, as the following way (Figure 7):

a. selecting a pattern of the method library according to a task type of a knowledge system,

b. concreting knowledge type of input, output and reference by using the software object ontology and the obtained business model as the requirement specification,

c. adding a control structure to the description with the obtained information of knowledge type,

d. selecting a pattern for each unit function of the description and continuing the above process.

Finally, we've provide 22 methods on the abstract-pattern level, 92 methods including prolog-build-in functions on the program-code level, and 69 methods on the middle-level.

4.4 Experimental Study

In order to consider the validness and usability of proposed framework, we've implemented the above mechanism by JAVA into the proto-typing tool and applied it into case studies of constructing business applications from description documents.

In each case study, some patent texts obtained from the Internet are used as a business document. We have compared between the models of case studies provided by Process Handbook and the ones built by the proto-type tool (Figure 7).

Roughly speaking about the result, approximately 70 % activities of each case study model are determined from patent texts, and, each implementation has been completed around 18 hours after receiving the patent text of claims and details as specifications.

The above means that the cost of the application development has been reduced more than 50% as compared with by hand, that reuse of the system has been performed about

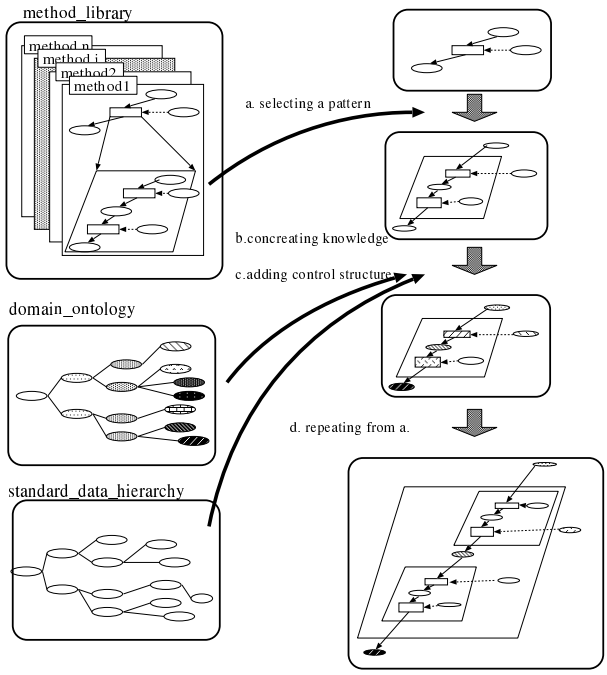


Figure 7: Overview of a Development Process

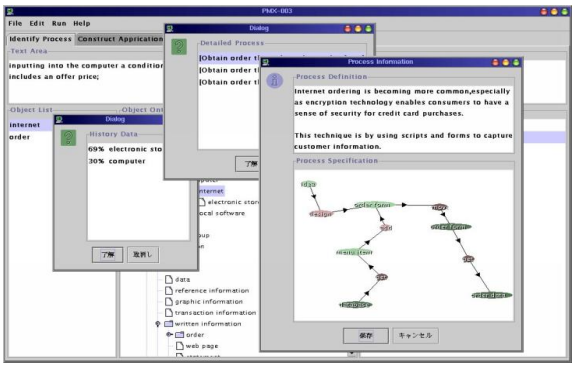


Figure 8: Execution of Supporting Tool

the common structure of business applications, and that main business structure could be reused if we have stacked and open some experience to the public at our library. Now, we are analyzing and investigating about the deeper experimental studies on the development on heterogeneous systems and the result will be open until the conference.

5 Discussions

The above result of the experimental study means that the cost of the application development has been reduced more than 50% as compared with by hand, that reuse of the system has been performed about the common structure of business applications, and that main business structure could be reused if we have stacked and open some experience to the public at our library. Now, we are analyzing and investigating about the deeper experimental studies on the development on heterogeneous systems.

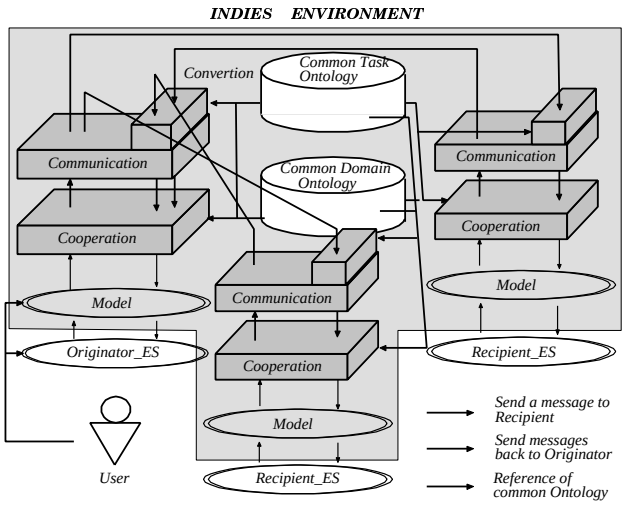


Figure 9: An Overview of INDIES

As comparison with related work, there are three main fields of research areas: clarifying specifications, building an application and reusing existing libraries.

First, as discussed in Introduction, numbers of work on analyzing business specification has been done by MIT, Edinburgh University and so on. Their work are very significant as a fundamental research, however, most of them are around abstract and general framework. Recently, Process Handbook is revised into e-Business Process Handbook and provides a hundred of specification as case studies. But most of them are just defined by natural language text. So, our work can be regarded as the integrated work to utilize the related work.

Second, a lot of works are carried out on building applications Including software engineering field[Code, et.al. 1997; **?**], but they use the different type of tool and languages on the different phases of development. So, models and languages should be unified into on framework as we proposed.

Our framework is based on a standpoint that it is difficult to automate the whole business, but possible to do many part of it. Recently, a few of the researchers consider the modeling methodology of the whole enterprise structure as a multi-agent system[Kendall 1999]. It is worth paying attention but still remain on the abstract structures of defining agent roles.

Finally, a lot of projects concentrate on reusing libraries, ontologies and applications and provide a number of repositories. One of our previous works is on interoperation of the heterogeneous expert systems[Izumi et.al. 1999a]. Because each expert system is modeled by its own vocabulary, it needs a conversion facility so that it can understand the messages sent from other expert systems. In the work, we employ a specification-sharing(SS)-based cooperation, called assisted coordination[Genesereth 1994]. The shared specification comes from REPOSIT library which serves as a common structure of noun and verb concepts named the common domain ontology and the common task ontology. As the methods of modeling, operationalizing, cooperating and communicating (wrapping) distributed expert systems come up, we put them together into an interoperation environment for distributed expert systems INDIES(Fig. 9).

In our previous work, a number of significant lessons obtained in exchanging the messages among the heterogeneous expert systems. However the way to construct the common ontologies is still remain as the future work. Our current work can be regarded as a new approach to reuse heterogeneous repositories based on ontologies.

6 Conclusion

As conclusion, the computing environment, which supports dynamic construction of business models and applications from business document, should be developed for the purpose to perform the re-engineering business processes according to the changes of business situations. From standpoint that the heterogeneous repositories should be integrated to achieve the unified support of the application development, we have proposed the framework of the extraction of the required information based on ontologies with reusable repositories such as e-business process handbook, Common KADS and REPOSIT. In order to construct the business models and to implement them as the actual business including software applications, we develop two repositories on different-grain-levels: the business specification repository on the level of business activities and the business software repository on the level of software applications.

We have implemented the prototype system by JAVA and confirmed that it supports us in various phases of business application development including business model manifestation, detailed business model definition and an implementation of business software applications. Furthermore, we are re-organizing our product in order to open it to the public.

References

[Code, et.al. 1997] P.Code, D.North, M.Mayfield, “Object Models:Strategies, Patterns, & Applications”, Yourdon Press, 1997.

[Fellbaum 1998] C.Fellbaum ed: “Wordnet”, The MIT Press, 1998.

[Fowler 1997] M.Fowler, “Analysis Patterns: Peusable Object Models”, Addison-Wesley, 1997.

[Genesereth 1994] M.R.Genesereth and S.P.Ketchpcl: Software Agents, CACM.ol.37.No.7. (1994) 48–53.

[Izumi et.al. 1999a] N.Izumi, A.Maruyama, A.Suzuki, T.Yamaguchi: “An Interoperative Environment for Developing Expert Systems” Proc. EKAW’99 LNAI.1621, pp.335–340, Springer–Verlag (1999).

[Izumi et.al. 1999b] N.Izumi, A.Maruyama, A.Suzuki, T.Yamaguchi: “Design and Implementations of Reusable Method Library for Development of Expert Systems”, Journal of JSAI, 14, 6, 1061–1071, (1999), in Japanese.

[Izumi and Yamaguchi 2000] N.Izumi, T.Yamaguchi: “Developing Software Agents Based on Product Ontology and Process Ontology”, Proc. ECAI-00 Workshop on Applications of Ontologies and Problem–Solving Methods (2000) 8-1—8-6.

[Kendall 1999] E.A.Kendall, “Role Models: Patterns of Agent Analysis and Design”, British Telecom Journal Special Issue on Decentralized Business Systems, (1999).

[MIT Process Handbook Project] The MIT Process Handbook Project: http://ccs.mit.edu/ph

[Petrie 1996] C.J.Petrie: “Agent-Based Engineering, the Web and Intelligence” IEEE Expert, 1996. URL: http://java.standord.edu/

[Schreiber et.al. 1999] Guuns Schreiber, et al: Knowledge Engineering and Management: The CommonKADS Methodology,MIT Press (1999).

[Uschold et.al. 1998] M.Uschold, et al: The Enterprise Ontology, Knowledge Engineering Review,Vol.13,Special Issue on Putting Ontologies to Use(1998).

[Yamaguchi 1999] T.Yamaguchi: Constructing Domain Ontologies Based on Concetp Drift Analysis, IJCAI Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends , 13-1 - 13-7, (1999.8)

[Common KADS] See http://www.commonkads.uva.nl

Engineering Ontologies using Semantic Patterns

Steffen Staab, Michael Erdmann

Institute AIFB, University of Karlsruhe
D-76128 Karlsruhe, Germany
www.aifb.uni-karlsruhe.de/WBS/
Ontoprise GmbH, Haid-und-Neu-Strasse 7
D–76131 Karlsruhe, Germany
www.ontoprise.com

Alexander Maedche

FZI Research Center
for Information Technologies
Haid-und-Neu-Strasse 10-14
D–76131 Karlsruhe, Germany
www.fzi.de/wim

Abstract

Interoperability is one of the major design objectives when building applications for B2B and Semantic Web applications. In this paper, we present a methodology for engineering semantic knowledge such that these semantic structures are easier reusable when switching between several representation languages. For this purpose, we reconsider the commonalities of representation languages and their usage in actual applications. Out of this consideration we derive *semantic patterns* as a means to communicate knowledge at an epistemological level of representation and as a means for (partial) execution by any particular implementation of any representation language. The underlying method we propose combines the advantages of formal specification methods (where feasible) with informal, natural language explanations such as used in software engineering for design patterns.

1 Introduction

The Web tremendously changed the way companies do their business, because it provides cheap, easy and widely available transport for information. Now, the Semantic Web is about to let the Web mature from a technical platform that allows for the transportation of syntactic information to the communication of knowledge. The prime format for the latter is RDF (Resource Description Framework) and RDFS (RDF-Schema). RDF [25] was designed by its developers in the Web way, *i.e.* as a smallest common denominator that a lot of people can easily adhere to, only representing a light-weight object model (cf., e.g., [3]) with URIs and reification. *RDFS* [8] adds an additional layer to integrate some simple notions of classes, class inheritance, properties and property inheritance. While RDF(S)[1] certainly goes an important step into the direction of the "Semantic Web", it only provides a very lightweight, and thus extremely restricted, semantic language. Therefore, a number of proposals for languages and language extensions on top of RDF and RDFS are currently under development (cf. [14; 2; 11], which describe some of them). Given the large variety of logics in use in many systems nowadays and given experiences from knowledge representation and reasoning[2] that have shown the necessity of this multitude of languages, the variety of these proposals gives only a first impression of the Babel of languages which will come up in the Semantic Web. This Babel, however, is counterproductive to semantic interoperability which lies at the heart of doing smart B2B on the Semantic Web (e.g., for exchanging knowledge about catalogues or about resource availability). This paper is about engineering machine-processable knowledge in a way such that it is reusable across different Semantic Web languages and across different styles of modeling.

Even before the wide-spread usage of the Web, there have been efforts to find one representation level for all languages (cf., KIF [20; 19]) and to automatically translate between different languages (cf., OntoLingua [22]), but both approaches heavily suffered from the fact that the *meaning* of representations, *i.e.* their semantic entailments, could not be adequately represented in a single *lingua franca*. In order to allow for reuse of semantic information and a multitude of underlying representation languages, we approach the problem from a different angle, an engineering point of view, by considering differences and commonalities of various languages at an explicitly modeled *epistemological level* (cf. [7]). We opt for, first, building on RDF(S) and, second, by constructing *semantic patterns* that capture the intended semantic entailments.

While RDF(S) allows for a frame model that virtu-

[1] We use "RDF(S)" to refer to the combined technologies of RDF and RDF-Schema.

[2] Various applications request different types of languages and reasoning systems, ranging from description logics systems (e.g., for data warehouse quality [17]), over — tractable — non-monotonic reasoning systems (e.g., non-monotonic inheritance for insurance help desk [27]), or systems that include temporal reasoning (e.g., for corporate history analysis [4]).

ally everyone may agree one, what really distinguishes and what is common to any two representation languages are the differences and commonalities of the semantic entailments expressible there. We show in this paper how to model commonalities in, what we call, *semantic patterns*. Semantic patterns are used for communication between Semantic Web developers on the one hand, but also for mapping and reuse to different target languages on the other hand, thus bridging between different representations and different ways of modeling knowledge. Developing the semantic patterns, we do not invent the wheel from scratch, but we pick insights from software engineering and knowledge representation research and integrate them for use in the Semantic Web.

By sheer principle, we cannot produce an exhaustive list of possible semantic patterns or show how the epistemological level should look like given any set of representation languages. Hence, we substantiate the claims we make with a case study considering as target representation systems OIL/FaCT [14], currently the most prominent semantic layer on top of RDF(S), and SiLRi [13], an F-Logic-based [24] representation system.

Outline of the paper. In the following, we start with our model for semantic patterns, their underlying rationale as well as their formal and informal components (Section 2). Then, we show how this model fits into the Semantic Web (Section 3), *i.e.* how it can be realized in RDF and from which point it has to evolve now. Subsequently, we illustrate our approach with a case study. We describe an application scenario, where semantics are brought to bear in a target representation independent way.

2 Semantic Patterns

The rationale and conceptual model of our approach is explained in this section. In order to give the broad view necessary to understand the problem of modeling knowledge for a variety of representation languages through semantic patterns, this section

1. analyses the abstract properties of our problem, thereby recollecting the most relevant related work in this area;
2. characterizes the high-level solution to the problem, which leads to informal means for communicating a semantic pattern together with formal descriptions of consistency constraints; and, finally,
3. defines these two aspects of semantic patterns exemplifying them by one extremely simple and useful pattern, for which no modeling primitive exists in most modeling languages and frameworks.

2.1 The Problem and some of its History

When one tries to reuse semantics across boundaries stemming from the usage of different representation languages, different actual representation tasks and their correspondingly different formal models, one may recognize characteristics of the semantic models that remain constant. Striving for semantic interchangability the crucial point lies in capturing these characteristics. This is difficult, because:

- Different language definitions come with different formal models. In general, the models of two different languages are not comparable at all. Thus, when one defines a translation there may not exist a criterion to evaluate the correctness of the translation.
- There may be several semantically equivalent statements in one language. Their equivalence is, in general (e.g., for first-order predicate logic), undecidable. Hence, their fully automatic translation is, in general, not feasible.
- Some choices for representation are not semantically motivated, but are made in order to generate some particular behaviour of the actual system.

Therefore, direct translations from one representation language into the next do not seem to yield a viable way. As a way around this dilemma, we consider the engineering task of constructing a particular representation. Rather than working hard to implement intended semantic entailments in statements of one particular — for the purpose of reuse and translation even arbitrary, language — the engineer may decide to explicitly model semantic entailments at a meta-level, instantiate the meta-level description, and compile the final representation into the one or the other target language.

In fact, those semantic entailments that are most widely agreed upon, such as necessary inheritance conditions, directly show up in common representation languages, such as `rdfs:subclass` and `rdf:type` in RDF(S). This also is the reason that subsequently we may easily assume that ground RDF facts are translatable into virtually all target languages.

Then, there is another medium size set of such semantic characteristics that are widely deemed interesting, that can be modeled independently from particular target languages in a number of systems and that can also be mapped into a wide range of languages. They are widely known from object-oriented databases. Gruber defined a set of primitives that captures them in his *Frame Ontology* [22] for use of comparatively simple translation between several representation languages (e.g., transitivity of relations, database joins, or disjointness of concepts).[3]

[3] Subsets of them have also been discussed/are under discussion for usage in description logics languages like OIL.

For characteristics more sophisticated than those mentioned above, there exists no comprehensive concept for engineering semantics in a way that is really reusable across several languages.

By its very nature, the problem of describing formal model characteristics for all representation languages is an open one that cannot be solved by producing a closed list of modeling primitives like the ones in Gruber's Frame Ontology. Hence, there is a need for a technique of describing new semantic primitives at a higher level of abstraction.

Again looking back into history, Brachman [7] and others have captured particular model characteristics, i.e. *semantic entailments*, in *axiom schemata* for the purpose of easier engineering of large sets of axioms. Axiom schemata provide an abstraction of actual axioms and particular axiom schemata were categorized and named. Doing so, Brachman introduced the name *epistemological level* for this layer of description. The results of his efforts were a set of epistemological primitives for description logics. Unlike our purpose, his goal was not the reuse of semantics across represenation languages, but rather the reuse of engineering efforts in one language.

2.2 The High-level Solution to the Problem

While axiom schemata already go into the direction of abstracting from formal model characteristics, by definition they are developed for one language only. Hence, one part of our high-level idea was to allow for (an open list of) **new epistemological primitives** that can be instantiated in different representation languages for modeling particular semantic entailments and which are, thus, similar to named axiom schemata working in one language.

However, one needs a more flexible paradigm better suited to apply to a larger range of representation languages and more able to abstract from particular formal models. As described above, the general problem does not allow to come up with a completely formal and ubiquitously translatable specification of semantics. Hence, the other part of our high-level idea is to require extra efforts from Semantic Web developers. To support them in their efforts, it appeared to be a prerequisite that they could communicate more efficiently about these new epistemological primitives — similar to the way that software engineers talk about recurring software designs.

Design Patterns and Semantic Patterns. Design patterns have been conceived for object-oriented software development to provide *(i)* a common design vocabulary, *(ii)* a documentation and learning aid, and *(iii)* support for reorganizing software. Likewise to the naming and cataloguing of algorithms and data structures by computer scientists, design patterns are used by software engineers to communicate, document and explore design alternatives by using a common design vocabulary or a design pattern catalog. By this way, they also decrease the complexity of developing and understanding of software systems. Additionally, design patterns offer solutions to common problems, help a novice "acting" more like an expert and facilitate the reverse-engineering of existing systems.

Though bridging between formal representations seems to be a formal task only, very often quite the contrary becomes true. When not everything, but only relevant aspects of knowledge can or need to be captured, when not all inferences, but only certain strains of semantic entailments can or need to be transferred, the development of new semantic primitives should not only allude to the formal definition of translations into target languages, but also to informal explanations. Therefore a semantic pattern does not only comprise new epistemological primitives, but likewise to design patterns, it also serves as a means for communication, cataloguing, reverse-engineering, and problem-solving. Thus, it may contribute to a more efficient exploitation of Semantic Web techniques.

Semantic Patterns and Consistency. Experiences in the related field of problem solving methods (cf. Section 6) have shown that there are as many interpretations of natural language descriptions as there are readers [15]. Given the preliminary that we do not want to subscribe to any particular, extremely powerful, and hence undecidable, specification language, we nevertheless need some means to describe consistency conditions that an implementation of a semantic pattern must adhere to.

The basic idea here is the following: A semantic pattern enforces semantic entailments on ground facts. A semantic pattern is implemented by translating its instantiated epistemological primitives into the target language. Thus, if one gives an instantiation of a semantic pattern together with some example ground facts related to the pattern, the implementation (i.e., the translation together with the target system) may derive semantic consequences. A translation may be considered consistent, if it derives those consequences out of the example ground facts that the developers of the semantic patterns wanted to be derived (i.e. the positive examples) and not those that they explicitly excluded (i.e. the negative examples).

This definition of *consistency of translations* is easy to realize, since it only builds on premises that are already given within the semantic patterns framework sketched so far. In particular, the translation of ground RDF facts into the target language is sometimes trivially done by an identity function (e.g., for OIL or DAML-Ont). Otherwise it is not overly complicated, because the RDF model already is a kind of least common denominator for the representation languages we consider.[4] The reader may note that this notion of con-

[4]The only counterexamples we could come up with were

sistency may not completely prevent misuse or misunderstanding. For instance, translations that map everything to the empty set always incur consistency without doing any good.

A complete description of semantic patterns including a formal specification of consistency will be given in the following.

2.3 Two Complementary Aspects of Semantic Patterns

This subsection puts the high-level rationale outlined above into a concrete perspective. We start with the informal description of the template structure of semantic patterns. Subsequently, we specify the formal parts of semantic patterns including consistency conditions for the translation into target representations.

Informal Description of Semantic Patterns. A Semantic Pattern consists of two major parts. The first part describes core elements that are completely independent from any actual implementation. The second part specifies example implementations, including descriptions about target system/language-specific behaviour.

The first part consists of the following eight core elements (also cf. Example Pattern part 1 given in Table 1):

1. **Pattern Name**: describes in few words the semantic problem, its solutions and consequences. Naming extends the pattern catalog and extends the semantic pattern vocabulary.
2. **Intent**: is a short statement describing what the pattern does and what its rationale and intent are.
3. **Also Known as**: enumerates other well-known names of this patterns (synonym list).
4. **Motivation**: describes a scenario that illustrates the semantic problem and elucidates how the semantic pattern may help in making implicit knowledge explicit.
5. **Structure**: represents the pattern. In particular, it gives the defining namespace, lists the relevant (new) epistemic primitive(s) and describes their signature.
6. **Known Uses**: shows examples of the pattern found in real Semantic Web applications.
7. **Related Patterns**: lists a number of closely related patterns (e.g. generalization hierarchy of patterns) and describes how they are related
8. **Constraints**: lists tuples of RDF facts and instantiated epistemic primitives ($C_{i,\text{in}}$, $C_{i,\text{out}}$, $C^{opt}_{i,\text{out}}$, $C_{i,\text{notout}}$, $C^{opt}_{i,\text{notout}}$). Their intended meaning is that for all i, given $C_{i,\text{in}}$, and thus using the new epistemic primitives of the pattern, $C_{i,\text{out}}$ *must* be derived in any given implementation and $C_{i,\text{notout}}$ *must not* be derived in any given implementation. In addition, one may include sets $C^{opt}_{i,\text{out}}$ and $C^{opt}_{i,\text{notout}}$ that, correspondingly, *should* and *should not* be derived in any given implementation.

The second part deals with implementation aspects of a Semantic Pattern. It consists of an arbitrary number of descriptions that relate the semantic pattern to particular target languages/systems. Each singleton entry (referring to one target language/system) should include the following five template elements (also cf. Example Pattern part 2 described in Table 2):

1. **Name of target language/system**: refers to the actual language specification. Because various system implementations may even incur different behavior (ranging from response time to various degrees of covering a specification), we also allow to specify system implementations rather than just language versions.
2. **Applicability**: The applicability of a semantic pattern in an actual language/system may be restricted. Example restrictions may necessitate the generation of new symbols in a particular target system or they may restrict the semantic entailments generated from $C_{i,\text{in}}$ to some subset of $C_{i,\text{out}}$ to mention but two example restrictions.
3. **Translation result of input constraints** $C_{i,\text{in}}$: shows the representation of an example fact base in the target language. Thus, the user of the Semantic Pattern sees an explicit example result of the translation process.
4. **Translation — Sample Speci`cation** : This specification describes the translation of instantiated epistemic primitives of the given pattern into the target language. The specification may be given in logics, pseudo code or a real programming language. In some target languages (e.g., F-Logic) it is reasonable to specify the translation in the target language itself. If the translation is given by a formal specification it can be considered to represent the translation T_i, which is referred to in the following.
5. **Comment**: Additional comments on particularities of the translation and the translation results.

The reader may note in the example templates that the various fields of the semantic patterns are not required. For ease of presentation, we have used italics in the running example to abbreviate redundant syntactic descriptions.

rather esoteric schemes, like monadic predicate logics.

[5] For better readability we here mostly use a PL1-style of denotation (without quantifiers) that can be easily mapped to RDF.

Table 1: Example Semantic Pattern — Part 1

Semantic Pattern	
Pattern Name	`Locally Inverse Relation`
Intent	Allows to define inverses the scope of which is restricted to particular concepts.
Also Known As	Restricted inverse relation
Motivation	Often the definition of global inverses is too generic and yields overly general inferences. For example, one may have ontology definitions that every MOVIE ISSHOWN in a THEATRE and every PLAY ISGIVEN in a THEATRE and THEATRE HOST EVENT. Now, the local inverse of ISSHOWN is HOST restricted to the range MOVIE and the local inverse of ISGIVEN is HOST restricted to the range PLAY. A global inverse might lead to unwanted consequences. For this reason this pattern allows the definition of inverse properties restricting their domain and range concepts. This notion of locality is naturally given in OO systems, where properties are defined locally in classes. It is not given in RDF(S) where properties are first-class citizens and exist independent of classes.
Structure	
Namespace	`http://ontobroker.aifb.uni-karlsruhe.de/schema/LocalInverse.rdf`
Epistemic Primitive(s)	LOCALINVERSE
Signature	LOCALINVERSE(r_1, c_1, r_2, c_2) with r_1, r_2 denoting binary relations and c_1, c_2 denoting their corresponding ranges
Known Uses	`http://ka2portal.aifb.uni-karlsruhe.de`
Related Patterns	The pattern "globally inverse relation" subsumes "locally inverse relation" when applied to the same relations
Constraints[5]	
$C_{1,\text{in}}$	LOCALINVERSE(ISSHOWN, THEATRE, HOST, MOVIE), DOMAIN(ISSHOWN, MOVIE), RANGE(ISSHOWN, THEATRE), DOMAIN(HOST, THEATRE), RANGE(HOST, EVENT), TYPE(Lassie, MOVIE), HOST(Schauburg, Lassie)
$C_{1,\text{out}}$	ISSHOWN(Lassie, Schauburg)
$C^{\text{opt}}_{1,\text{out}}$	TYPE(Schauburg, THEATRE)
$C_{1,\text{notout}}$	ISGIVEN(Lassie, Schauburg), TYPE(Lassie, PLAY)

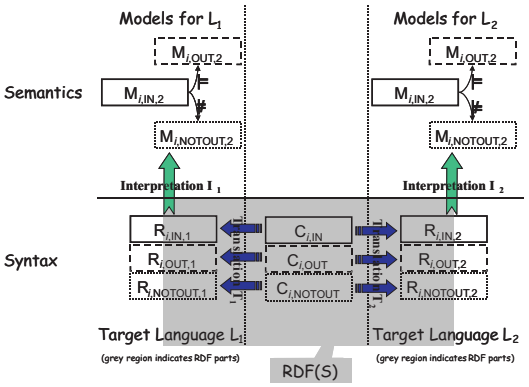


Figure 1: Checking for consistency

Formal Consistency Conditions for Semantic Patterns. Combining our considerations on consistency with our actual specification of semantic patterns we may now describe the overall setting in formal terms (also cf. Figure 1).

We base our consistency checking on facts in RDF (cf. $C_{1,\text{in}}, C_{1,\text{out}}, C_{1,\text{notout}}$ in Figure 1), which may include some of the new epistemic primitives. Each translation maps this RDF based representation into a target language (or system) L_j, resulting in target representations $R_{1,\text{in},j}, R_{1,\text{out},j}, R_{1,\text{notout},j}$ $(j = 1, 2)$. From any consistent translation T_j the interpretation of output facts $R_{1,\text{out},j}$ must and the interpretations of facts $R_{1,\text{notout},j}$ must not be semantically entailed by the corresponding interpretation of input constraints $R_{1,\text{in},j}$. Depending on the actual system, semantic entailment ($\models$) or not-entailment ($\not\models$) may be replaced by syntactic derivation ($\vdash$) or not-derivation ($\not\vdash$).

To describe this intuition precisely, we specify:

De`nition 1 (Translation Mapping) *A translation mapping is any function* $T_i : 2^{\textit{sentences(RDF)}} \rightarrow 2^{\textit{sentences}(L_i)}$, *where sentences*$(X)$ *stands for all legal sentences of language* X, L_i $(i = 1 \ldots n)$ *are representation languages, and* $2^{\textit{sentences}(X)}$ *describes the set of all subsets of all legal statements of language* X.

This simply boils down to: a translation mapping for target language L_i is able to translate every possible RDF representation into this target language.

In this definition we assume that new epistemological primitives are defined by statements in RDF (also cf. Section 3). Given such a translation mapping, we

Table 2: Example Semantic Pattern — Part 2

Semantic Pattern Implementations for Locally Inverse Relations	
Language/System	OIL/FaCT
Applicability	Requires the creation of artificial relations for this type of modeling.
Translation (Sample Code)	copy RDF literally, create two new subproperties with specialized range restrictions and declare appropriate INVERSE relation
Translation Result for $C_{1,\text{in}}$	*literal copy of the statements in $C_{1,in}$ plus RDF equivalent of ...* **slot-def** *host1* **subslot-of** *host* **inverse** *isShown* **slot-def** *host2* **subslot-of** *host* **inverse** *isGiven*
Comment	The reader may note that in contrast to `rdfs:subPropertyOf` OIL's **subslot-of** allows for cycles.
Language/System	F-Logic/SiLRi
Applicability	Applicable.
Translation (Sample Code)	translate RDF syntactically and add two meta-rules (see below)
Translation Result for $C_{1,\text{in}}$	*Syntactic translation of statements in $C_{1,in}$ plus ...* $FORALL\ C1, C2, R1, R2, O1, O2$ $O2[R2 \twoheadrightarrow O1]\ and\ O1 : C1 \leftarrow$ $\text{LOCALINVERSE}(R1, C1, R2, C2)\ and\ O1[R1 \twoheadrightarrow O2]\ and\ O2 : C2.$ $FORALL\ C1, C2, R1, R2, O1, O2$ $O1[R1 \twoheadrightarrow O2]\ and\ O2 : C2 \leftarrow$ $\text{LOCALINVERSE}(R1, C1, R2, C2)\ and\ O2[R2 \twoheadrightarrow O1]\ and\ O1 : C1.$
Language/System	Predicate Logic
Applicability	Applicable.
Translation (Sample Code)	*add the following PL2 Specification* $FORALL\ C1, C2, R1, R2, O1, O2$ $R2(O2, O1) \wedge \text{TYPE}(O1, C1) \leftarrow$ $\text{LOCALINVERSE}(R1, C1, R2, C2) \wedge \text{TYPE}(O2, C2) \wedge R1(O1, O2)$ $FORALL\ C1, C2, R1, R2, O1, O2$ $R1(O1, O2) \wedge \text{TYPE}(O2, C2) \leftarrow$ $\text{LOCALINVERSE}(R1, C1, R2, C2) \wedge \text{TYPE}(O1, C1) \wedge R2(O2, O1)$

can evaluate to which degree it is consistent with regard to the constraint specification of the given semantic pattern.

De`nition 2 *Let the semantic pattern S include the constraints ($C_{i,in}$, $C_{i,out}$, $C^{opt}_{i,out}$, $C_{i,notout}$, $C^{opt}_{i,notout}$) for $i := 1 \ldots m$. A translation mapping T_j is called consistent with the Semantic Pattern S iff forall $i := 1 \ldots m : T_j(C_{i,in}) \models T_j(C_{i,out})$ and $T_j(C_{i,in}) \not\models T_j(C_{i,notout})$.*

De`nition 3 *Let the semantic pattern S include the constraints ($C_{i,in}$, $C_{i,out}$, $C^{opt}_{i,out}$, $C_{i,notout}$, $C^{opt}_{i,notout}$) for $i := 1 \ldots m$. A translation mapping T_j is called strongly consistent with the Semantic Pattern S iff T_j is consistent with S and forall $i := 1 \ldots m : T_j(C_{i,in}) \models T_j(C^{opt}_{i,out})$ and $T_j(C_{i,in}) \not\models T_j(C^{opt}_{i,notout})$.*

3 Semantic Patterns for the Semantic Web

Building on the rational and methodology outlined above, the basic idea of semantic patterns on the Web has two major dimensions: First, there is the dimension of technical representation and, second, there is the social process of establishing Semantic Pattern libraries on the Web.

3.1 Representing Semantic Patterns in RDF

Semantic patterns are used for communicating some information to human developers and some information to computer systems. Hence, RDF is also the ideal format for the representation of the Semantic Pattern itself.

We have provided a RDF-Schema description for semantic patterns available at `http://ontoserver.aifb.uni-karlsruhe.de/schema/PatternSchema.rdf` which describes RDF resources of `rdf:type` *Pattern*.

We refer to this schema with the namespace prefix ps for Pattern Schema. An actual instantiation of this schema, *viz.* our running example, locally inverse relation, is shown at http://ontoserver.aifb.uni-karlsruhe.de/schema/LocalInverse.rdf. For easier presentation we refer to it in the following outline of these RDF structures by the namespace prefix pa for Pattern Application.

A semantic pattern is a rdfs:Resource and can be associated with other resources by a number of defined properties. The properties ps:patternName, ps:intent, ps:alsoKnownAs, and ps:motivation have been described in Section 2 and associate a pattern object with literal, textual values (rdf:parseType="Literal"). The property ps:relatedPattern links a pattern to related ones. The structure of a pattern is modeled by the two properties ps:epistemicPrimitive and ps:signature. The former represents a simple literal while the latter associates a pattern with several named properties (e.g. pa:class1 or pa:rel2). These properties define the parameters of the pattern and must define the pattern itself as their rdf:domain. The ranges of the parameter properties represent the parameter types for the pattern. The example pattern has four parameters: pa:class1 and pa:class2 of type rdfs:Class, and pa:rel1 and pa:rel2 of type rdf:Property.

All mentioned information becomes a part of the actual pattern description, i.e. the RDF model. Applications can ask for the signature of a pattern by querying this model, esp. the ps:signature and ps:epistemicPrimitive properties of the pattern. This formal part of the model can directly be exploited for further processing, e.g. for building GUIs for instantiating a particular semantic pattern, e.g. for instantiating the locally inverse relations-pattern with HOST, MOVIE, ISSHOWN, and THEATRE.

The constraints ($C_{i,\text{in}}$, $C_{i,\text{out}}$ etc.) used for checking consistency of actual implementations represent partial models that are only true within their consistency checking context, but not on a global scale. The means of RDF for representing contextual information is *reification*. Therefore the different constraint sets are modeled via reification. Each set of constraint statements is retrievable from a pattern-resource by querying one of the constraint-properties. Each such property relates pattern-resources with rdf:Statements. Translation functions (cf. Definition 1) may access these sets and translate the reified statements into the target language.

The second part of semantic patterns as described in Section 2 defines target language/system-specific information about the pattern, of possible implementations, and expected results of translation functions. The description of the name of the target language and system are modeled as literal values of the properties ps:language and ps:system, respectively. The translation code may be stored within another RDF-literal accessible via the ps:code property of the ps:Implementation resource. Results of applying this code to the sample given in the constraint set $C_{i,\text{in}}$ can be represented in the RDF-model of the implementation as well. Since languages exist that directly operate on the RDF-model, it is possible to store reified RDF-statements reachable via the ps:C_in_rdf-property. Applications that do not understand RDF syntax may retrieve the transformation of the statements $C_{i,\text{in}}$ from ps:C_in_literal.

It is our general policy to allow developers a lot of leeway. Currently, all mentioned properties are optional and in the typical case either ps:C_in_rdf or ps:C_in_literal is given but not both.

The reader may note that the formal description (in RDF) of formal parts allows for direct digestion of constraints and signatures for aims such as code generation, consistency checking, and user interface construction.

3.2 Semantic Pattern Libraries

Eventually, the need for particular semantic patterns is driven by Semantic Web developers. With the engineering of ontologies on the Web (cf., e.g., [1]) new ideas will come up about what type of inferencing shall be supported and, hence, made interchangable between representation systems.

Since this development is in its infancy right now, we have started to collect a number of semantic patterns that seem widely applicable:

- Gruber's Frame Ontology includes a set of over 60 primitives, some of which are found in core RDF(S), e.g. rdf:type, and some of which are somewhat more sophisticated, e.g. symmetry of relations or composition (database joins).
- Medical knowledge processing often relies on the engineering of *part-whole reasoning* schemes such as appear or do not appear when we consider the following examples: *(i)*, the appendix is part of the intestine. Therefore, an appendix perforation is an intestinal perforation. And, *(ii)*, the appendix is part of the intestine, **but** an inflammation of the appendix (appendicitis) is not an inflammation of the intestine (enteritis).

  We have described how to represent structures that allow for expressing (for *(i)*) and preventing (for *(ii)*) these semantic entailments in RDF in [29] — in a preliminary version of the semantic patterns framework.
- Inheritance with exception is a semantic pattern that is very often useful. Its application and its tractable, even efficient, technical reasoning part has been described, e.g., in [27]. The core idea is that one considers the inheritance of properties, allows for the non-inheritance

of certain properties, and uses a particular, unambiguous strategy for resolving conflicts between paths of inheriting and non-inheriting a particular property. A simple example is that a PATIENT's treatment may be covered by medical insurance, a NON-COMPLIANT PATIENT's treatment may not be covered, but a NON-COMPLIANT, MENTALLY DISTURBED PATIENT's treatment will be paid by the insurance company. Hence, coverage of treatment is typically inherited, e.g. by almost all subclasses of patient, but not by ones like NON-COMPLIANT PATIENTs.

Note that often there is no translation into particular target languages for this pattern. For instance, it can be realized in Prolog or F-Logic, but not in the standard description logics systems.

- A number of patterns may be derived from object-oriented or description logics systems, e.g. *local range restrictions* are very often useful. A simple example is that the *parentOf* a HUMAN is restricted to HUMAN, the *parentOf* a FOX is restricted to FOX, while the range restriction of *parentOf* may be ANIMAL in general.

A more complete elaboration of these and other patterns is currently under development. In particular, we investigate how software engineering methodology about modeling and code generation from an evolving library of semantic patterns can be brought to bear within our modeling environment (cf. Section 5).

4 Using Semantic Patterns — A Case Study

In [28] we have described how "Semantic Community Web Portals" using ontologies can be built. The ontology acts as a semantic backbone for accessing information on the portal, for contributing information, as well as for developing and maintaining the portal. We discussed a comprehensive and flexible strategy for building and maintaining a high-value community web portal, where the development and maintenance process consists of the stages *requirements elicitation*, *web site design*, *ontology engineering* and *query formulation*. The reasoning service for the portal was provided by SiLRI [13], which is essentially based on F-Logic [24; 12], only ground facts may alternatively be provided in RDF syntax. F-Logic fits nicely with the structures proposed for RDF and RDFS, however, F-Logic does not offer any support for interoperability of representation mechanisms, *i.e.* axioms written in F-Logic and the implicit knowledge that comes from applying them to the fact base are extremely hard to reuse in other representation frameworks. In this section we show how our approach fits with a recent proposal for representing knowledge on the web, namely OIL, the ontology inference layer [14]. OIL, which is in several semantic respects "orthogonal" to F-Logic, offers inferencing [23] on a semantic layer on top of RDF(S).

In the following case study we show the usage of semantic patterns for meeting the needs of an actual application, while allowing for the engineering of semantics on a level that is transportable to OIL *and* F-Logic (and many other representation schemes). The case study described here relies on the tools and techniques we employed for building "Semantic Community Web Portals". We here consider a **Cultural Event Portal**, that integrates distributed information from movie databases and cinema programs and offers semantic access to the information provided.

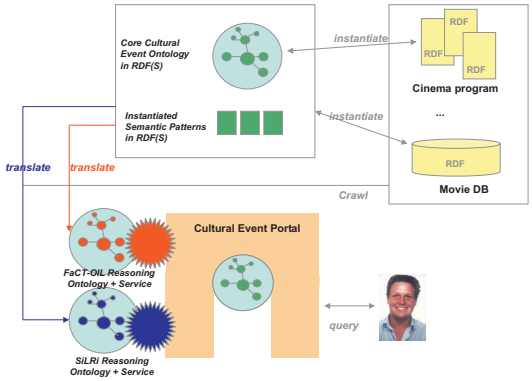


Figure 2: Case Study — Building a Semantic Cultural Event Portal

Figure 2 depicts the overall framework. Based on the core ontology, actual facts are generated at the information provider side. For building a semantic cultural event portal with sophisticated reasoning we additionally instantiate semantic patterns on top of the core ontology. The core ontology with instantiated semantic patterns may be translated into OIL and/or F-Logic. Facts are crawled from the information providers and given to the reasoning services. The portal accesses the underlying reasoning services and provides comprehensive information on cultural events.

In the following we give some examples how the core ontology looks like, show how actual semantic patterns are defined and translated into OIL and/or F-Logic.

4.1 Modeling the Core Ontology

We use our Ontology Engineering Environment OntoEdit (cf. Section 5 and Figure 3) for engineering class and property definitions in RDF(S) with graphical means. Parts of the core ontology are given as follows:

```
<rdfs:Class rdf:ID="Event"/>
<rdfs:Class rdf:ID="Movie">
 <rdfs:subClassOf rdf:resource="Event"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Play">
 <rdfs:subClassOf rdf:resource="Event"/>
</rdfs:Class>
 <rdfs:Class rdf:ID="Theatre"/>
<rdf:Property rdf:ID="host">
 <rdfs:domain rdf:resource="Theatre"/>
```

```
 <rdfs:range rdf:resource="Events"/>
</rdf:Property>
<rdf:Property rdf:ID="isShown"/>
 <rdfs:domain rdf:resource="Movie"/>
 <rdfs:range rdf:resource="Theatre"/>
</rdf:Property>
<rdf:Property rdf:ID="isGiven"/>
 <rdfs:domain rdf:resource="Play"/>
 <rdfs:range rdf:resource="Theatre"/>
</rdf:Property>
```

The ontology defines the conceptual backbone for generating RDF metadata on the information provider side, as for example given through the following statements:

```
<cultev:Movie rdf:ID="movie:Lassie">
 <cultev:name>Lassie</cultev:name>
 <cultev:hasActor
   rdf:resource="actor:RoddyMcDowall"/>
</cultev:Movie>
```

4.2 Generating an OIL ontology with Semantic Patterns

An OIL ontology is built on top of the core RDF(S) data model and contains descriptions of classes, slots and individuals [14]. Classes are unary predicates and may be related to other classes by stating that one is a subclass of another. Slots are binary relations, they may also be related to each other via the notion of subslots.

In our example application, Cultural Event Portal, additional reasoning on top of core RDF(S) is required. We therefore instantiate some patterns on top of the core RDF(S) ontology to enforce semantic constraints and then translate them into the more powerful OIL. In our scenario we use two patterns, namely the *local range restriction* and the *locally inverse relations* pattern. The pattern *local range restriction* (cf. Section 3.2) adds to the class definition of MOVIE with respect to the property ISSHOWN the range restriction THEATRE. The following statements are added within the concept definition of MOVIE:

```
<oil:hasSlotConstraint>
 <oil:ValueType>
  <oil:hasProperty rdf:resource="isShown"/>
  <oil:hasClass rdf:resource="Theatre"/>
 </oil:ValueType>
</oil:hasSlotConstraint>
```

In Section 2 our mechanism for defining semantic patterns has been introduced using the example of *locally inverse relations* patterns. OIL offers the definition of global inverses, that is often too generic and yields overly general inferences. In our example, we defined in our ontology that every MOVIE ISSHOWN in a THEATRE and every PLAY ISGIVEN in a THEATRE and THEATRE HOST EVENT. Now, the local inverse of ISSHOWN is HOST restricted to the range MOVIE and the local inverse of ISGIVEN is HOST restricted to the range PLAY.

As OIL does not directly support locally inverse relations, the creation of artificial relations is required. The translation into OIL is given through the following statements:

```
<rdf:Property rdf:ID="host1">
 <oil:subSlotOf rdf:ID="host"/>
 <rdfs:domain rdf:resource="Theatre"/>
 <rdfs:range rdf:resource="Movie"/>
</rdf:Property> <rdf:Property rdf:ID="host2">
 <oil:subSlotOf rdf:ID="host"/>
 <rdfs:domain rdf:resource="Theatre"/>
 <rdfs:range rdf:resource="Play"/>
</rdf:Property>
<rdf:Property rdf:ID="isShown">
 <oil:inverseRelationOf rdf:resource="host1"/>
</rdf:Property>
```

We introduce two new properties HOST1 and HOST2 as subslots of HOST [6]. The range of property HOST1 is restricted to the MOVIE class, the range of property HOST2 is restricted to the PLAY class. Additionally we use the `inverseRelationOf` construct of OIL to denote that the property ISSHOWN is inverse to the property HOST1.

We also give the translation of `locally inverse relation` pattern to Frame-Logic. The pattern is applicable and is generated via the F-Logic statements we have seen before:

LOCALINVERSE(ISSHOWN, MOVIE, HOST, THEATRE).
$FORALL\ C1, C2, R1, R2, O1, O2 \quad O2[R2 \twoheadrightarrow O1]\ and\ O1 : C1 \leftarrow$
LOCALINVERSE$(R1, C1, R2, C2)\ and\ O1[R1 \twoheadrightarrow O2]\ and\ O2 : C2.$
$FORALL\ C1, C2, R1, R2, O1, O2 \quad O1[R1 \twoheadrightarrow O2]\ and\ O2 : C2 \leftarrow$
LOCALINVERSE$(R1, C1, R2, C2)\ and\ O2[R2 \twoheadrightarrow O1]\ and\ O1 : C1.$

Essentially, this version was used for the Community Web Portal, but it could not be communicated to outsiders of F-Logic.

5 OntoEdit

Our general approach for engineering ontologies in conjunction with developing and using semantic patterns has been or is currently being implemented in ONTOEDIT [30], an ontology engineering workbench for building web ontologies[7]. In this section we give an outline of how an ontology engineering environment is augmented by components for realizing semantic patterns.

The modeling of the core ontologies builds on RDF(S) primitives. The process is started by collecting terms for classes and organizing them hierarchically; in

[6] We use the oil:subSlotOf component as defined in the denotational semantics of standard OIL available at http://www.cs.man.ac.uk/ horrocks/OIL/Semantics/oil-standard.html.

[7] More detailed information can be obtained at http://ontoserver.aifb.uni-karlsruhe.de/ontoedit.

parallel one may add properties to the ontology. Several different views for building the ontology are offered to the user. Figure 3 depicts the graphical user interface of OntoEdit: On the left hand side of Figure 3 the class hierarchy of our cultural event ontology is depicted. The class-property view offers the user the possibility to attach properties to classes. Properties may also be defined globally and organized hierarchically.

OntoEdit offers a number of predefined semantic patterns. On the lower right part of figure 3 the interface for instantiating global inverseness and locally restricted inverseness is depicted. The user selects properties and defines their (local) inverses explicitly. If the user also restricts domain and range of the properties the semantic pattern `locally inverse relations` is instantiated. The text descriptions of the semantic patterns are available in OntoEdit's help.

Once conceptual modeling is completed, one may use OntoEdit to explore the defined ontology including the newly instantiated semantic patterns. For this purpose, one may crawl example RDF facts, translate the semantic patterns into F-Logic or OIL and then explore ontology and facts by querying the test examples.

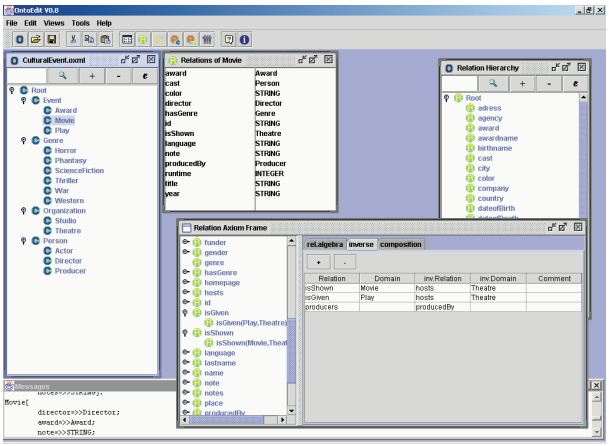


Figure 3: Snapshot of OntoEdit Web Ontology Workbench

6 Related Work

This paper is motivated by the need to share and exchange semantic knowledge on the Web (cf., e.g., [9] for general motivation or [28] for an actual application). This need comprises the integration of various sources on the content level as well as on the representation level, i.e. integrating knowledge from various basic representation mechanisms available (like [8]) or on the rise (like [14; 2]).

We started out from the area of *ontology engineering* aiming at conceptual models that could be used in multiple underlying representations (cf. [29]). Doing so, we extended related work in the field of knowledge representation using *axiom schemata*.Because our goal was not only to formally represent, but to allow for rich communication between developers who create actual implementation based on various representation systems, we looked into software and knowledge engineering dealing with *design and knowledge patterns and problem-solving methods*.

6.1 Ontology Engineering & RDFS

In our earlier proposals [30] we have discussed how to push the engineering of ontological axioms from the *symbol level* onto the *knowledge level* — following and extending the general arguments made for ODE [6] and Ontolingua [16]. Also similar to our RDF(S)-based ontology engineering tool OntoEdit is Protégé [21], which provides comprehensive support for editing RDFS and RDF, but lacks any support for axiom modeling and inferencing. In contrast to all of these approaches, we aim also at *partial* descriptions of semantic entailments such as very often necessary when switching from one to the other representation paradigm.

6.2 Axiom Schemata

The usage of axiom schemata in various paradigms has been a major motivation for our approach (cf. Subsection 2.3). In particular, we have relied on experiences with engineering axiom schemata in F-Logic and on related work that exploits axiom schemata in various description logics dialects.

F-Logic. The logical model of F-Logic, essentially a rich model for datalog, fits nicely with the structures proposed for RDF and RDFS. This also led to the first implementation of an inference engine for RDF (SiLRi [13]). SiLRi provides many inferencing possibilities one wants to have in RDF and, hence, has provided an excellent start for many RDF applications. In fact, it even allows to use axioms in restricted second-order logic, but these axioms may not be denoted in RDF, but only directly in F-Logic.

Description Logics. Description logics has been derived from efforts for specifying the axiom schemata that are most relevant for terminological engineering. Hence, its development provides valuable input for relevant semantic patterns such as the ones exploited in our case study (cf. Section 4). To speak more precisely, description logics constitutes not a single, but a set of similar languages. A large amount of research has been undertaken to explore the effects of adding additional syntactic and semantic features to existing versions of description logics. However, their efforts remain very far from bridging between mutually incompatible representation paradigms, which is the goal of our approach.

A web-compatible version of description logics has been presented with OIL [14]. OIL is intended as

a common core language that is more powerful than RDF, but is intended to provide a basic layer rather than a language "all singing all dancing". As our case study also illustrates, even OIL does not suffice for all potential needs, but semantic patterns may also be used on top of OIL — rather than "only" on top of RDF.

Combinations of Different Logics. Obviously, there has been the need for interoperability between F-Logic and Description Logics and, hence, Levy and Rousset [26] proposed an integration of a (simple) Description Logics approach with horn rules. In the end, however, neither one of them nor their integration will be sufficient for all possible purposes and applications of the future Semantic Web. A similar statement holds for current combinations of modal logics; in fact, the field as a whole is very young and can be exploited for practical purposes only to very limited extent in the near future (cf. the excellent survey paper [5]). Along similar lines KIF [19] was invented, but was most often only used at the syntactical rather than at the semantic level of knowledge transportation. Building on KIF, Gruber [22] has investigated the translation between languages using the frame ontology as its interlingua. Though the frame ontology is very useful (essentially it catches the primitives used in object-oriented database systems), the language is too restricted in general.

We have shown in this paper, how to use semantic patterns with OIL, a Web-compatible description logics framework and F-Logic, a language that had been intensively used for Semantic Web applications [28]. Thereby, our semantic patterns are not restricted to either of these paradigms or their integration.

6.3 Patterns and Problem Solving Methods

Design patterns [18] — and their knowledge engineering counterparts [10] — have proved extremely successful in describing characteristics of the *contents* that are to be described (algorithmic structures or knowledge structures). We in contrast have focused on the description of *language characteristics* in order to bridge between different representation languages, thus applying the paradigms of patterns at the meta-level.

A similar contrast holds between semantic patterns and problem solving methods. "Problem solving methods describe domain-independent reasoning components" [15]. They come at various levels of abstractions, from informal text, over few lines of pseudocode up to implementations in a particular language. Problem solving methods can be thought of as a variety of search methods with heuristics that benefit from domain specific knowledge where the heuristic is built into the problem solving method itself.

While on the very high level problem solving methods may appear similar to semantic patterns, there are several major distinctions, only two of which we want to mention here: First, semantic patterns only describe *what* needs to be inferenced they do not specify how semantic entailments are actually derived in a particular representation, which is the domain of problem solving methods that describe *how* to do things. Second, the domain proper of semantic patterns and problem solving methods is rather dissimilar. Typical problem solving method libraries include, e.g., "propose and revise", or "heuristic classification", while semantic patterns such as we propose abstract from language characteristics to include, e.g., "part-whole reasoning", "local inverses", or "inheritance with exceptions".

7 Conclusion

We have shown a new methodology, *viz. semantic patterns*, for engineering semantics on the Web in a way that makes it easier to reuse in a wide range of existing representation systems and easier to communicate between different Semantic Web developers. Semantic patterns are used to describe intended semantic entailments and, thus, allow a higher level of abstraction above existing Semantic Web languages — similarly as software design patterns allow to abstract from actual applications.

With this approach, there comes now the possibility to bridge between various paradigms for representation. By semantic patterns, the social process of designing new and communicating previously successful semantic patterns may now be started. The reader, however, may bear in mind that semantic patterns only provide a ground of discourse for man and machine. Which actual patterns will eventually turn out to be successful for which purpose will have to be shown over time by the Web community.

References

[1] Daml ontology library. http://www.daml.org/ontologies/, observed 2000.

[2] Daml-ont initial release. http://www.daml.org/2000/10/daml-ont.html Observed at October 12, 2000, 2000.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query language for semi-structured data. *Journal of Digital Libraries*, 1(1):68–88, 1997.

[4] J. Angele, H.-P. Schnurr, S. Staab, and R. Studer. The times they are a-changin' — the corporate history analyzer. In D. Mahling and U. Reimer, editors, *Proceedings of PAKM-2000. Basel, Switzerland, October 30-31, 2000*, 2000.

[5] B. Bennett, C. Dixon, M. Fisher, E. Franconi, I. Horrocks, U. Hustadt, and M. de Rijke. Combinations of modal logics. submitted for publication, 2000.

[6] M. Blázquez, M. Fernández, J. M. García-Pinar, and A. Gómez-Pérez. Building ontologies at the knowledge level using the ontology design environment. In *In Proceedings of KAW-98, Banff, Canada, 1998*, 1998.

[7] R. Brachman. On the epistomological status of semantic networks. *Associative Networks*, pages 3–50, 1979.

[8] D. Brickley and R.V. Guha. Resource description framework (RDF) schema specification. Technical report, W3C, 1999. W3C Proposed Recommendation. http://www.w3.org/TR/PR-rdf-schema/.

[9] V. Christophides and D. Plexousakis, editors. *Proceedings of the ECDL-2000 Workshop — Semantic Web: Models, Architectures and Management*, 2000.

[10] P. Clark, J. Thompson, and B. Porter. Knowledge patterns. In A. Cohn, F. Giunchiglia, and B. Selman, editors, *In Proc. of KR-2000, Breckenridge, CO, USA, 12-15 April 2000*, pages 591–600, San Mateo, CA, 2000. Morgan Kaufmann.

[11] O. Corby, R. Dieng, and C. Hebert. A conceptual graph model for w3c resource description framework. In *In Proceedings of ICCS-2000. Darmstadt, Germany, August 2000*, LNAI. Springer, 2000.

[12] S. Decker. On domain-specific declarative knowledge representation and database languages. In *Proc. of the 5th International KRDB Workshop*, pages 9.1–9.7, 1998.

[13] S. Decker, D. Brickley, J. Saarela, and J. Angele. A query and inference service for RDF. In *QL'98 - The Query Languages Workshop*. W3C, 1998. http://www.w3.org/TandS/QL/QL98/.

[14] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the web. In *Proceedings of the DL-2000, Aachen, Germany*, 2000.

[15] D. Fensel and E. Motta. Structured development of problem solving methods. *IEEE Transactions on Knowledge and Data Engineering*, to appear.

[16] R. Fikes, A. Farquhar, and J. Rice. Tools for assembling modular ontologies in Ontolingua. In *Proc. of AAAI 97*, pages 436–441, 1997.

[17] E. Franconi and G. Ng. The i.com tool for Intelligent Conceptual Modeling. In *Proceedings of 7th International KRDB Workshop. Berlin*. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-29/.

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns — Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[19] M. R. Genesereth. Knowledge interchange format. draft proposed american national standard (dpans). ncits.t2/98-004. http://logic.stanford.edu/kif/dpans.html seen at Sep 7, 2000, 1998.

[20] M. Ginsberg. Knowledge interchange format: the KIF of death. *AI Magazine*, 5(63), 1991.

[21] E. Grosso, H. Eriksson, R. W. Fergerson, S. W. Tu, and M. M. Musen. Knowledge modeling at the millennium — the design and evolution of Protégé-2000. In *In Proceedings of KAW-99, Banff, Canada, 1999*, 1999.

[22] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[23] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, pages 636–647, 1998.

[24] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[25] O. Lassila and R. Swick. Resource description framework (RDF). model and syntax specification. Technical report, W3C, 1999. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax.

[26] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.

[27] L. Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103:1–34, 1998.

[28] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic community web portals. In *In Proc. of WWW9, Amsterdam, The Netherlands, May, 15-19, 2000*. Elsevier, 2000.

[29] S. Staab, M. Erdmann, A. Maedche, and S. Decker. An extensible approach for modeling ontologies in RDF(S). In Christophides and Plexousakis [9].

[30] S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In *In Proceedings of the ECAI-2000 Workshop on Ontologies and Problem-Solving Methods. Berlin, August 21-22, 2000*, 2000.