On-To-Knowledge Methodology — Final Version

York Sure, Rudi Studer
(University of Karlsruhe)

Executive Summary.
On-To-Knowledge EU IST-1999-10132 Project Deliverable D18 (WP5)
This deliverable is the final version of the OTK methodology. Core contributions of this document are: (i) An overview of the On-To-Knowledge building blocks and their relationships, (ii) a methodology for introducing and maintaining ontology based knowledge management solutions into enterprises with a focus on Knowledge Processes and Knowledge Meta Processes and, last but not least, (iii) the illustration of process steps by examples and lessons learned derived from applying the OTK tool suite in the OTK case studies according to the methodology.

Document Id.	OTK/2002/D18/v1.0
Project	On-To-Knowledge EU IST-1999-10132
Date	26th September, 2002
Distribution	Restricted

On-To-Knowledge Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-1999-10132. The partners in this project are: Vrije Universiteit Amsterdam VUA (coordinator, NL), Institute AIFB / University of Karlsruhe (Germany), Schweizerische Lebensversicherungs- und Rentenanstalt/Swiss Life (Switzerland), British Telecommunications plc (UK), CognIT a.s. (Norway), EnerSearch AB (Sweden), AIdministrator Nederland BV (NL).

Vrije Universiteit Amsterdam (VUA)
Division of Mathematics and Informatics W&I
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 4447718, Fax: +31 20 872 27 22
Contactperson: Dieter Fensel
E-mail: dieter@cs.vu.nl

Schweizerische Lebensversicherungs- und Rentenanstalt / Swiss Life
Swiss Life Information Systems Research Group
General Guisan-Quai 40
8022 Zürich
Switzerland
Tel: +41 1 284 4061, Fax: +41 1 284 6913
Contactperson: Ulrich Reimer
E-mail: Ulrich.Reimer@swisslife.ch

CognIT a.s
Busterudgt 1.
N-1754 Halden
Norway
Tel: +47 69 1770 44, Fax: +47 669 006 12
Contactperson: Bernt. A.Bremdal
E-mail: bernt@cognit.no

AIdministrator Nederland BV
Julianaplein 14B
3817 CS Amersfoort
The Netherlands
Tel: +31 33 4659987, Fax: +31 33 4659987
Contactperson: Jos van der Meer
E-mail: Jos.van.der.Meer@aidministrator.nl

University of Karlsruhe
Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 3923, Fax: +49 721 608 6580
Contactperson: Rudi Studer
E-mail: studer@aifb.uni-karlsruhe.de

British Telecommunications plc
BT Adastral Park
Martlesham Heath
IP5 3RE Ipswich
United Kingdom
Tel: +44 1473 605536
Fax: +44 1473 642459
Contactperson: John Davies
E-mail: John.nj.Davies@bt.com

EnerSearch AB
Carl Gustafsväg 1
SE 205 09 Malmö
Sweden
Tel: +46 40 25 58 25; Fax: +46 40 611 51 84
Contactperson: Hans Ottosson, Fredrik Ygge
E-mail: hans.ottosson,fredrik.ygge@enersearch.se

Sirma AI, Ltd. - Artificial Intelligence Labs, Onto-Text Lab.
38A Christo Botev blvd
Sofia 1000, Bulgaria
Tel: (+359 2) 981 23 38
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Contents

Chapter 1

Introduction

Availability of electronically stored information increased drastically through the development of the World Wide Web. Currently it contains more than a billion documents, but support for accessing and processing information is limited. Most information is only presentable but not understandable by computers. Tim Berners-Lee envisioned the Semantic Web (Berners-Lee et al., 2001; Fensel et al., 2002) that aims at providing automated access to information due to machine-processable semantics of data. Ontologies (Fensel, 2001) formalize a shared understanding of a domain (Uschold & Grueninger, 1996) and therefore play a crucial role for communication among human beings and software agents.

The EU-IST-1999-10132 On-To-Knowledge project develops and explores sophisticated methods and tools for Knowledge Management and thereby provides infrastructure for the Semantic Web. It's partners include technology providers who build an ontology based tool environment to support the acquisition, maintenance and access of information, technology users who evaluate and use the tool environment in industrial case studies and a methodology provider who guides and assists during the application of the tool environment in the case studies.

This deliverable (deliverable 18, workpackage 5) is the final version of the OTK methodology, which is a process oriented methodology for introducing and maintaining ontology based knowledge management solutions into enterprises. It integrates and extends well-known methodologies from the field of knowledge engineering. Real-world experiences from the case studies illustrate the methodological steps. The methodology is accompanied by the OTK tool suite, together they were applied in the case studies.

The document intends to give a complete overview of the project, core contributions of this document are therefore:

- An overview of the On-To-Knowledge building blocks and their relationships, *e.g.* including the overall project setting of OTK tool suite, case studies and methodology,
- a methodology for introducing and maintaining ontology based knowledge management solutions into enterprises with a focus on Knowledge Processes and Knowledge Meta Processes and, last but not least,
- the illustration of process steps by examples and lessons learned derived from applying the OTK tool suite in the OTK case studies according to the methodology.

This document is structured as follows:

Chapter 2 describes On-To-Knowledge building blocks. We will firstly zoom into knowledge management issues relevant for the project in Section 2.1 and introduce the notions of "Knowledge Items" and "Meta data" in Section 2.2. Then, in Section 2.3, we briefly show the historical roots of "Ontology" and motivate the recently very popular usage of ontologies in computer science. Finally, we present and overview of the On-To-Knowledge project setting including (i), in Section 2.4, the OTK case studies, (ii), in Section 2.5, the OTK technical architecture including the OTK tool suite and OIL, and (iii), in Section 2.7, the role of this OTK methodology within the project.

Chapter 3 starts by a discussion of relevant processes for implementing and inventing knowledge management (KM) applications in Section 3.1, *i.e.* "Knowledge Meta Process", "Human Resource Management" and "Software Engineering". The focus in this work is on the Knowledge Meta Process, we depict the Knowledge Meta Process in Section 3.2 and subsequently illustrate each of the steps, *i.e.* the "Feasibility Study" in Section 3.3, "Kickoff" in Section 3.4, "Refinement" in Section 3.5, "Evaluation" in Section 3.6 and "Application & Evolution" in Section 3.7. The Knowledge Meta Process guides and supports the initial set up of an ontology based KM application.

Chapter 4 illustrates the Knowledge Process that focusses on using an KM application. It starts in Section 4.1 with depicting the process steps and describing each of the steps in detail in the following sections, *i.e.* "Knowledge Creation" in Section 4.2, "Knowledge Import" in Section 4.3, "Knowledge Capture" in Section 4.4, "Knowledge Retrieval & Access" in Section 4.5 and "Knowledge Use" in Section 4.6.

We conclude in Chapter 5 by sketching the tackled problem in Section 5.1, presenting a summary of the contribution from On-To-Knowledge to solve that problem in Section 5.2, and, last but not least, giving an outlook to further research in Section 5.3.

Chapter 2

Building Blocks

In this chapter we will describe On-To-Knowledge building blocks. We will firstly zoom into knowledge management issues relevant for the project and thereby introduce the notions of "Knowledge Items" and "Meta data". Then, we briefly show the historical roots of "Ontology" and motivate the recently very popular usage of ontologies in computer science. Finally, we present and overview of the On-To-Knowledge project setting including (i) the OTK case studies, (ii) the OTK technical architecture, *i.e.* the OTK tool suite and OIL, and (iii) the role of this OTK methodology within the project.

2.1 Knowledge Management

In recent years Knowledge Management (KM) has become an important success factor for enterprises. Increasing product complexity, globalization, virtual organizations or customer orientation are developments that ask for a more thorough and systematic management of knowledge – within an enterprise and between several cooperating enterprises. Obviously, KM is a major issue for human resource management, enterprise organization and enterprise culture – nevertheless, information technology (IT) plays the crucial enabler for many aspects of KM. As a consequence, KM is an inherently interdisciplinary subject.

IT-supported KM solutions are built around some kind of organizational memory (Abecker et al., 1998) that integrates informal, semi-formal and formal knowledge in order to facilitate its access, sharing and reuse by members of the organization(s) for solving their individual or collective tasks (Dieng et al., 1999). In such a context, knowledge has to be modelled, appropriately structured and interlinked for supporting its flexible integration and its personalized presentation to the consumer. Ontologies have shown to be the right answer to these structuring and modeling problems by providing a formal conceptualization of a particular domain that is shared by a group of people in an organization (O'Leary, 1998; Gruber, 1995).

There exist various proposals for methodologies that support the systematic introduction of KM solutions into enterprises. One of the most prominent methodologies is CommonKADS that puts emphasis on an early feasibility study as well as on constructing several models that capture different kinds of knowledge needed for realizing a KM solution (Schreiber et al., 1999). Typically, these methodologies conflate two processes that should be kept separate in order to achieve a clear

identification of issues (Staab et al., 2001): whereas the first process addresses aspects of introducing a new KM solution into an enterprise as well as maintaining it (the so-called "Knowledge Meta Process"), the second process addresses the handling of the already set-up KM solution (the so-called "Knowledge Process") (see Figure 2.1). *E.g.* in the approach described in (Probst et al., 1999), one can see the mixture of aspects from the different roles that, *e.g.* "knowledge identification" and "knowledge creation" play. The Knowledge Meta Process would certainly have its focus on knowledge identification and the Knowledge Process would rather stress knowledge creation.

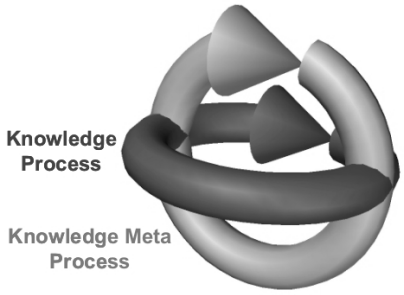

Figure 2.1: Two orthogonal processes with feedback loops

However, Knowledge Management is a process which is not only governed by IT. Hence, one needs to keep the balance between human problem solving and automated IT solutions. This balancing distinguishes KM from traditional knowledge-based systems. Nevertheless, the extensive knowledge modeling tasks that are inherent in ontology-based KM approaches support Alun Preece's saying "Every KM project needs a knowledge engineer".

2.2 Knowledge Items

The core concern of IT-supported knowledge management is the computer-assisted capitalization of knowledge (Abecker et al., 1998). Because information technology may only deal with digital, preferably highly-structured, knowledge the typical KM approach distinguishes between computer-based encoding in an organizational memory and direct transfer that is done by humans. Sticking to what is almost readily available, KM systems have tended to serve *either* the needs of easy access to documents (*e.g.* building on groupware, etc.) *or* the encoding of knowledge that facilitates the direct transfer of knowledge by humans (*e.g.* by people yellow pages, skill databases, etc.).

Introducing KM to a company (*i.e.* moving along the Knowledge Meta Process in "the light grey circle" in Figure 2.1), a very simple, pragmatic approach has typically been pursued, which however meant that only the low hanging fruits were picked. This approach is summarized in the left column of Table 2.1. What appears preemminent in this approach is the focus on the handling of documents (steps 2 and 3) and the existing, but minor role of the appendix "process". In spite of its immediate successes, this approach shows several disadvantages. In particular, it often leads to the consequence that the Knowledge Process steps ("the dark grey circle") of creation, import, capturing, retrieving/accessing, and using are only very loosely connected, if at all (*cf.* Figure 4.1).

Table 2.1: Approaching the Knowledge Process — two extreme positions

	Document focus	Knowledge Item focus
1.	Find out what the core knowledge needs are	
2.	Find out which *business documents* and *databases* deal with these knowledge needs	Find out which *knowledge items* deal with these knowledge needs
3.	Build an *Infrastructure* for your organizational memory system	Organize the *knowledge processes* to allow for creation, handling, and process support of and around knowledge items
4.	Re-organize *processes* to deal with creation and distribution of knowledge	Build an *Infrastructure* for your organizational memory system

The underlying reason is that for each of these steps different types of business documents play a major role, which makes "knowledge re-use" – and not only knowledge re-finding – extremely difficult.

Subsequently, we show how *domain ontologies* (see next section 2.3) may act as the glue between knowledge items, bridging between different Knowledge Process steps. Thereby, we argue for a refocus on the Knowledge Process and its core items, which need not be documents! This shift becomes visible in the second column of Table 2.1, which positions *knowledge items* and *Knowledge Processes* in the center of consideration.

The reader may note that we contrast two rather extreme positions in Table 2.1. As becomes obvious in recent research papers, current knowledge management research tends to move away from the document focus to a focus on knowledge items and processes (Abecker et al., 1998; Staab & O'Leary, 2000). While for a multitude of settings we still see the necessity for the document-oriented view, we argue for a more constructivist view of the Knowledge Processes. In particular, we believe that the exuberant exchange and trading of knowledge within and across organization still has to begin – and that it needs a knowledge item-oriented view such as we plead for.

Relevant knowledge items appear in a multitude of different document formats: text documents, spreadsheets, presentation slides, database entries, web pages, construction drawings, or e-mail, to name but a few. The challenge that one must cope with lies in the appropriate digestion of the knowledge, *e.g.* by "simple" reuse, or by aggregation, combination, condensation, abstraction, and by derivation of new knowledge from aggregations. Following only the lines of traditional document management, IT support for knowledge management cannot take advantage of the contents of the business documents, but only of its explicit or implicit classification. At the other extreme of this spectrum, there are expert systems that structure and codify all the knowledge that is in the system. Though such an approach may sometimes be appropriate, it is certainly not the way to follow in the typical knowledge management scenario, where not everything can be codified, a lot of knowledge is created sporadically, and the worth of knowledge re-use is only

shown over time and not necessarily obvious from the very beginning.

Hence, one must search for the adequate balance between reuse, level of formality, and costs to codify knowledge. For instance, certain helpdesk scenarios imply long term use of extremely well-defined knowledge items (Morgenstern, 1998). Then it may be worth to codify extensively and to spend some considerable amount of time and money on coding. On the other hand, a sporadic discussion is typically not worth coding at all, since it lives on the spur of the moment and often is negligible and, hence, not reusable after some short time.

As a way to balance these conflicting needs and to flexibly manage various degrees of encoded knowledge, we advertise the use of various notions of *meta data*. The different notions of the term "meta data", *i.e.* data about data, may be classified at least into the following categories:

1. Data describing other data. We may again divide this category into two orthogonal dimensions.
   (a) The one dimension concerns the formality of this data. Meta data may range from very informal descriptions of documents, *e.g.* free text summaries of books, up to very formal descriptions, such as ontology-based annotation of document contents.
   (b) The second dimension concerns the containment of the meta data. Parts of meta data may be internal to the data that is described, *e.g.* the author tag inside of HTML documents, while others may be stored completely independently from the document they describe, such as a bibliography database that classifies the documents it refers to, but does not contain them.
2. The second major connotation of meta data is data that describes the structure of data. For our purpose, one might refer to this notion by the term "meta meta data", because we describe the structure of meta data. Also, in our context this notion boils down to an ontology that formally describes the domain of the KM application, possibly including parts of the organization and the information structures (Abecker et al., 1998). The ontology allows to combine meta data from different parts of the Knowledge Process and data proper that adhere to the ontology description.

Meta data in its first connotation fulfills a double purpose. It condenses and codifies knowledge for reuse in other steps of the KM process by being connected through mutual relationships and the ontology (the meta meta data). Furthermore, it may link knowledge items of various degrees of formality together, thus allowing a sliding balance between depth of coding and costs.

One of the core ideas of On-To-Knowledge is to extract meta data semi-automatically from documents and subsequently to use the extracted meta data for querying and browsing. Before giving an overview of the On-To-Knowledge project setting itself, we will now explain more detailed the second connotation of meta data, *i.e.* the notion of "ontology". We will briefly show the historical roots and motivate the recently very popular usage of ontologies in computer science.

2.3 Ontology

"Ontology" is a philosophical discipline, a branch of philosophy that deals with the nature and the organization of being. The term "Ontology" has been introduced by Aristotle in Metaphysics, IV,

1. In the context of research on “Ontology”, philosophers try to answer questions “what being is?” and “what are the features common to all beings?”.

According to (Guarino, 1998) we consider the distinction between “Ontology” (with the capital “O”) , as in the statement “Ontology is a fascinating discipline” and “ontology” (with the lowercase “o”), as in the expression “Aristotle’s ontology”. The former reading of the term ontology refers to a particular philosophical discipline the latter term has different senses assumed by the philosophical community and the computer science community. In the philosophical sense we may refer to an ontology as a particular system of categories accounting for a certain vision of the world. Such a system does not depend on a particular language in the philosophical point of view.

In recent years ontologies have become a topic of interest in computer science (*cf. e.g.* (Gruber, 1995; Noy & Hafner, 1997; Maedche & Staab, 2001; Fensel, 2001)). In its most prevalent use in computer science, an ontology refers to an engineering artifact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary. Usually a form of first-order logic theory is used to represent these assumptions, vocabulary appear as unary and binary predicates, called concepts and relations, respectively. Both, vocabulary and assumptions, serve human and software agents to reach common conclusions when communicating.

2.3.1 Reference and Meaning

The general context of communication (with or without ontology) is described by the meaning triangle (Ogden & Richards, 1923). The meaning triangle defines the interaction between symbols or words, concepts and things of the world (*cf.* Figure 2.2).

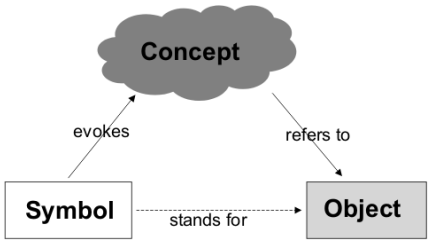

Figure 2.2: The Meaning Triangle

The meaning triangle illustrates the fact that although words cannot completely capture the essence of a reference (= concept) or of a referent (= thing), there is a correspondence between them. The relationship between a word and a thing is indirect. The correct linkage can only be accomplished when an interpreter processes the word invoking a corresponding concept and establishing the proper linkage between his concept and the appropriate thing in the world (= object).

2.3.2 Logics

Ontologies are usually expressed in a logic-based language, so that fine, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. Some ontology tools can perform automated reasoning using the ontologies, and thus provide advanced services to intelligent applications such as: conceptual/semantic search and retrieval, software agents, decision support, speech and natural language understanding, knowledge management, intelligent databases, and electronic commerce.

Figure 2.3 ((*cf.* (Maedche, 2002))) depicts the overall setting for communication between human and software agents. We mainly distinguish three layers: First of all, we deal with things that exist in the real world, including in this example human and software agents, cars, and animals. Secondly, we deal with symbols and syntactic structures that are exchanged. Thirdly, we analyze models with their specific semantic structures .

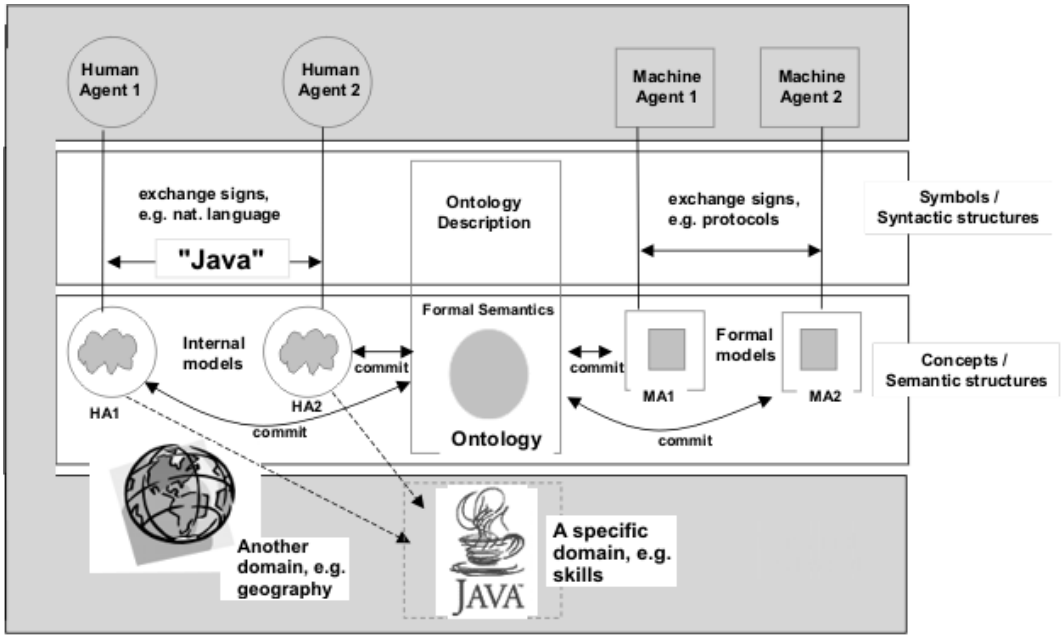

Figure 2.3: Communication between human and/or software agents

Let us first consider the left side of Figure 2.3 without assuming a commitment to a given ontology. Two human agents HA_1 and HA_2 exchange a specific sign, *e.g.* a word like "Java". Given their own internal model each of them will associate the sign to his own concept referring to possibly two completely different existing things in the world, *e.g.* the part of Indonesia called "Java" *vs.* the programming skill "Java". The same holds for software agents: They may exchange statements based on a common syntax, however, they may have different formal models with differing interpretations.

We consider the scenario that both human agents commit to a specific ontology that deals with a specific domain, *e.g.* skills. The chance that they both refer to the same thing in the world increases considerably. The same holds for the software agents SA_1 and SA_2: They have actual

knowledge and they use the ontology to have a common semantic basis. When agent SA_1 uses the term "Java", the other agent SA_2 may use the ontology just mentioned as background knowledge and rule out incorrect references, *e.g.* ones that let "Java" stand for the part of Indonesia. Human and software agents use their concepts and their inference processes, respectively, in order to narrow down the choice of references.

2.3.3 Classification of "ontologies"

Different classification systems for ontologies have been developed (van Heijst, 1995; Guarino, 1998; Jasper & Uschold, 1999). A classification system that uses the subject of conceptualization as a main criterion has been introduced by (Guarino, 1998). He suggests to develop different kinds of ontologies according to their level of generality as shown in Figure 2.4.

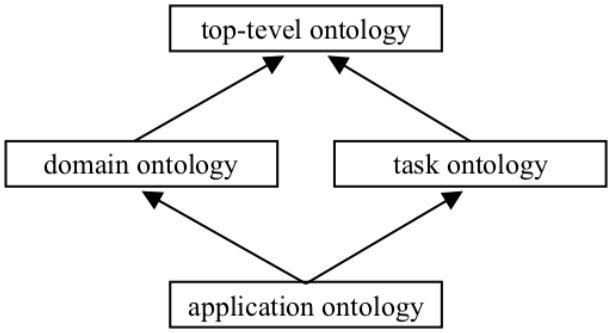

Figure 2.4: Different kinds of ontologies and their relationships

The following different kinds of ontologies may be distinguished as follows:

- **Top-Level ontologies** describe very general concepts like space, time, event, which are independent of a particular problem or domain. It seems reasonable to have unified top-level ontologies for large communities of users. Recently, these kinds of ontologies have been also introduced under the name "foundational ontologies".
- **Domain ontologies** describe the vocabulary related to a specific domain by specializing the concepts introduced in the top-level ontology.
- **Task ontologies** describe the vocabulary related to a generic task or activity by specializing the top-level ontologies.
- **Application ontologies** are the most specific ontologies. Concepts in application ontologies often correspond to roles played by domain entities while performing a certain activity.

In the On-To-Knowledge case studies the focus is set on application- and domain-ontologies that may be used in different kinds of ontology-based applications.

2.3.4 Some Naming Conventions

For this document we consider the following basic naming conventions and principle ideas. Typically, an ontology consists of *classes* which are organized in an "is-a" hierarchy known from object oriented languages (instead of "classes" one often sees the notion of "concepts"; an example for the hierarchy is "subclass *is-a* superclass"), *literals* (also known as "attributes", *e.g.* "a person has a name that is typically a textual string"), *properties* (also known as "relations", *e.g.* "a person works for a company") and *instances* (*e.g.* "'Dieter Fensel' is an instance of a person"). In our scenarios we rely on state-of-the art representation languages, *viz.* RDF(S) (Brickley & Guha, 1999) providing a basic ontology language and the extensions OIL (Fensel et al., 2001) and its successor DAML+OIL (DAML+OIL, 2001) adding additional logical structures. We made the experience from the case studies, that domain experts not familiar with ontology languages prefer the wording "concepts, attributes, relations and instances" instead of the wording introduced by RDF(S). Basically, we will stick to the above mentioned wording in this document and give an explanation if other wordings occur.

We will now illustrate the project setting by briefly introducing the OTK case studies, the technical architecture (including the OTK tool suite and the ontology language OIL) and the role of the methodology in this setting.

2.4 OTK Case Studies

Three case studies are carried out to evaluate and use the OTK tool environment for ontology based knowledge management. These case studies represent a broad spectrum of use cases. First, there are three industry sectors involved: insurance, telecom and energy. Second, the partners come from three countries with different cultures. Therefore they are facing various aspects of knowledge management problems.

2.4.1 Organizational Memory @ Swiss Life

Swiss Life (Switzerland) is a large insurance company serving customers around the world. Their vision is to build an organizational memory with an intranet based portal that offers a single entry point to the knowledge space of the company. The case study (Novotny & Lau, 2000; 2001; Novotny et al., 2001) explores different parts of their intranet:

1. Skills management (Lau & Sure, 2002; Sure et al., 2000) makes skills of employees explicit. Within the case study existing skill databases and documents (like *e.g.* personal homepages) are integrated and expanded. Two aspects are covered by the case study: first, explicit skills allow for an advanced expert search within the intranet. Second, one might explore its future career path by matching his current skill profile *vs.* job profiles.

   To ensure that all integrated knowledge sources are used in the same way, ontologies are used as a common mean of interchange to face two major challenges. Firstly, being an international company located in Switzerland, Swiss Life has internally four official languages, *viz.* German, English, French and Italian. Secondly, there exist several spellings of

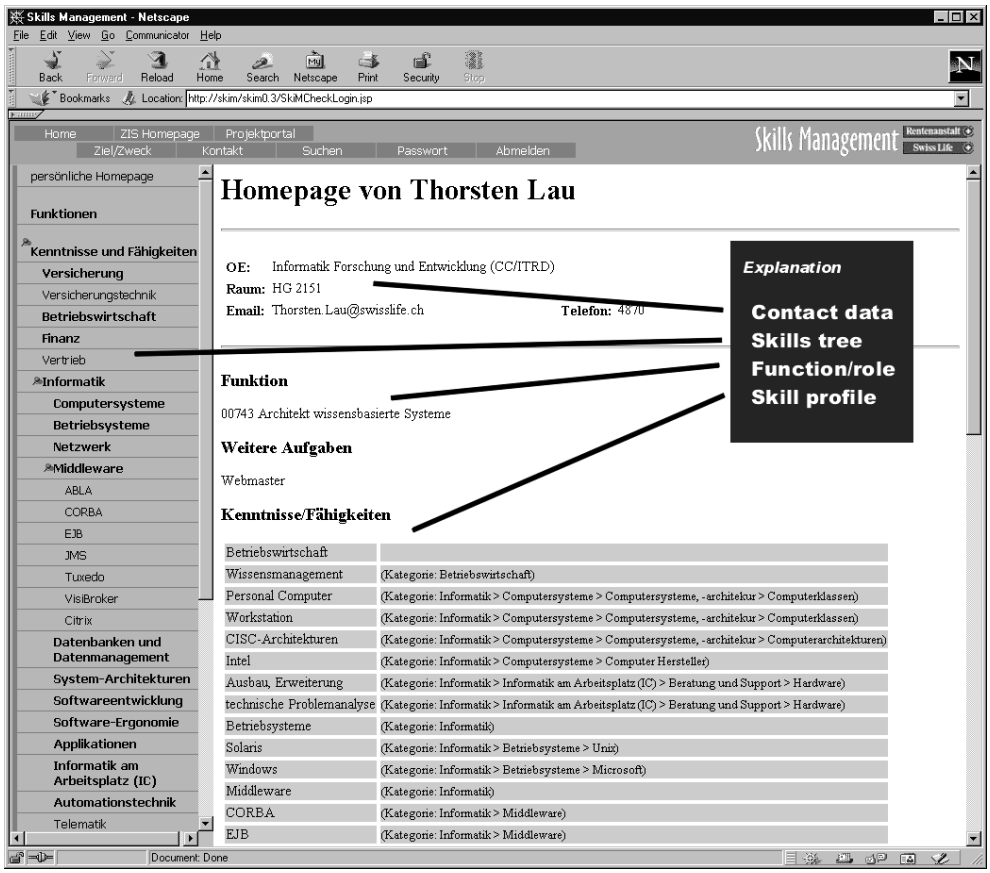

Figure 2.5: Skills management case study @ Swiss Life

same concepts, *e.g.* "WinWord" *vs.* "MS Word". To tackle these problems, ontologies offer external representations for different languages and allow for synonyms.

2. The International Accounting Standards (IAS) document is part of the global Swiss Life Intranet. The content of the document is highly specialized and even trained people hardly find relevant passages, even though there is a division into chapters and sections. Providing sophisticated access to the IAS is the second goal of this case study.

Figure 2.5 shows a screenshot from the skills management application. The prototype enables any employee to integrate personal data from numerous distributed and heterogeneous sources into a single coherent personal homepage. Any home page will contain:

- General contact information, such as functional unit, room, or telephone number (organizational data),
- other public descriptions, such as education, current position,
- details of personal skills, and

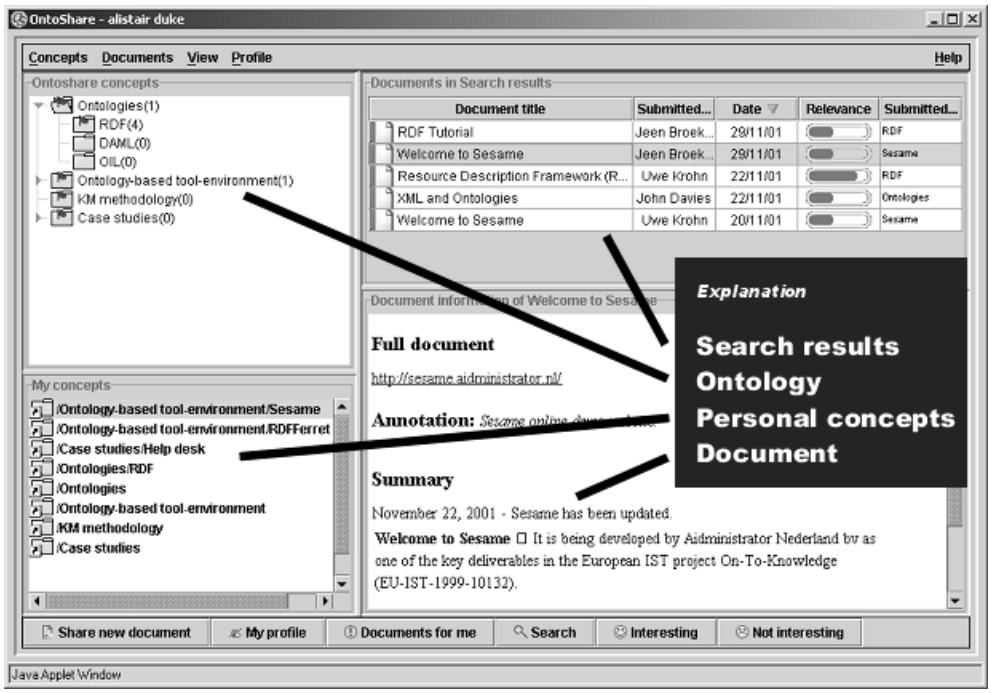

Figure 2.6: Communities of knowledge sharing with OntoShare @ BT

- a section that employees are free to fill in with personal interests, hobbies and categories, such as “I am a member of the professional associations...”, or “I am familiar with the following specialized literature...”.

2.4.2 Community of Knowledge Sharing @ BTExact Technologies

BT (United Kingdom) is a leading company on the telecom market and BTExact Technologies its subdivision that focusses on the development and application of new technologies. Knowledge sharing is regarded as an essential internal business process and therefore BTExact not only has a tradition in developing and selling knowledge sharing facilities, but also in applying them internally.

The case study was carried out within BT’s Research and Development organisation – BTexact Technologies. A group of people from BTexact who are researching into the fields of conferencing, knowledge management and personalisation were chosen. These people are researchers, developers and technical marketing professionals. Such individuals need to share knowledge as part of their job – they are often referred to as knowledge workers.

The goal of the case study[1] (Krohn & Davies, 2001; Duke & Davies, 2002b; 2002a) is to introduce the newly developed ontology based OntoShare system as a successor of the pre-existing knowledge sharing system (*cf.* Figure 2.6 for an example). As a major part this case study includes a user-focussed evaluation of OntoShare.

[1]Due to internal restructuring, the case study changed from a call center help desk scenario to the here described application scenario.

2.4.3 Virtual Organization @ EnerSearch

EnerSearch (Sweden) is a virtual organization founded by a consortium of major energy providers researching new IT-based business strategies and customer services in deregulated markets. Its research affiliates and shareholders are spread over many countries (*e.g.* US, Sweden, Germany ...). Essentially EnerSearch creates knowledge which is then transferred to shareholders and other interested parties. Goal of the case study (Iosif & Sure, 2002; Sure & Iosif, 2002; Iosif et al., 2001; Iosif & Ygge, 2002; Iosif & Mika, 2002) is to enhance the knowledge transfer to researchers in different disciplines and countries, and to specialists from shareholding companies interested in getting up-to-date information about R&D results on IT in Energy. Ontologies will help to enable a content based search on research topics. A special focus in this case study is on the user focussed evaluation of ontology based tools from OTK (QuizRDF and Spectacle) *vs.* typical keyword based retrieval (EnerSEARCHer) during an experiment.

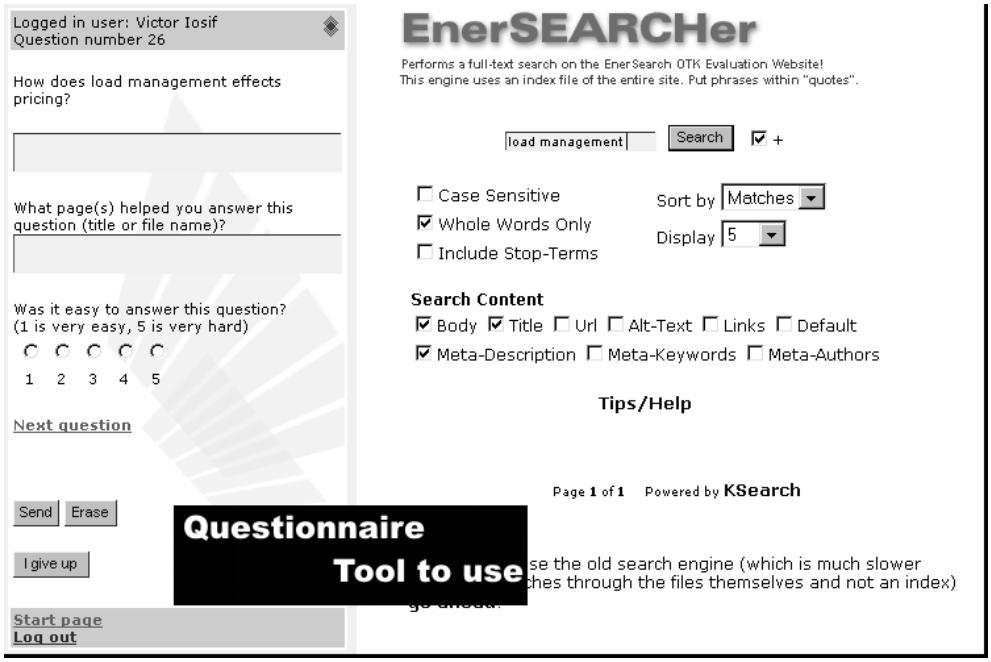

Figure 2.7: Virtual organization case study @ EnerSearch

Figure 2.7 shows the setting during the experiment (here shown for the free text search engine EnerSEARCHer, QuizRDF and Spectacle will be shown in figures 2.9 and 2.10). The screen is splitted into two parts, *i.e.* frames. (i) On the left side you see a frame that is used to guide users through the questionnaire during the experiment. After a user is logged into the system, he gets presented question by question in the upper part and can type in the answers. Along with each answer a user should also note how easy he found to answer a particular question (on a scale from 1–"easy" to 5–"very hard"). If a user could not find any answer he could push the "I give up button". Each user had to answer 30 questions, *i.e.* 10 with each tool. As mentioned in the paper, the questions were mixed up for different user groups. Which leads us to the second part, the tools themselves. (ii) On the right side the currently active tool for answering a question (EnerSEARCHer, QuizRDF or Spectacle) was presented to the user, *i.e.* the tool a user had to use

for answering a question was given in this frame. Here you see the GUI of EnerSEARCHer, where you can easily recognize a typical query interface for keyword based search engines.

2.5 OTK Technical Architecture

2.5.1 Tool Suite

A key outcome of the On-To-Knowledge project is the resulting software toolset[2]. Several consortium partners are participating in the effort to realize in software the underpinning ideas and theoretical foundations of the project.

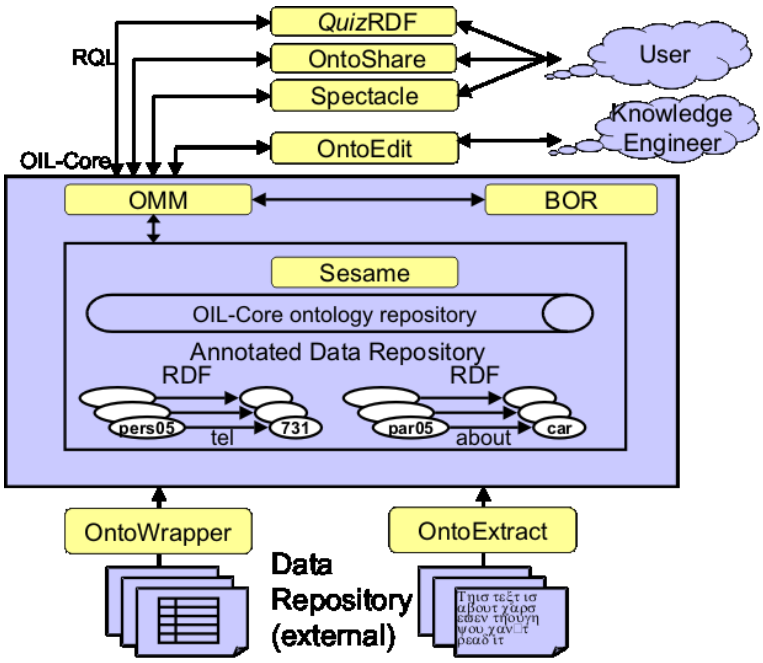

Figure 2.8: OTK technical architecture

A major objective of the project is to create intelligent software to support users in both accessing information and in the maintenance, conversion, and acquisition of information sources. The tools are integrated in a three-layered architecture (*cf.* Figure 2.8). The layers consists of (i) the user front end layer on top, (ii) a middleware layer in the middle and (iii) an extraction layer at the bottom. Each tool represents certain functionalities. The layering allows for a modular design of applications that bundle some or all of the functionalities provided. Most of the tools presented in the figure are described subsequently below. As a minimum requirement all tools support OIL core that has been designed to be exactly the part of OIL that coincides with RDF(S) (*cf.* section 2.5.2).

[2] A technical fact sheet provides technical requirements for the installation of the OTK tool suite (Sure, 2002).

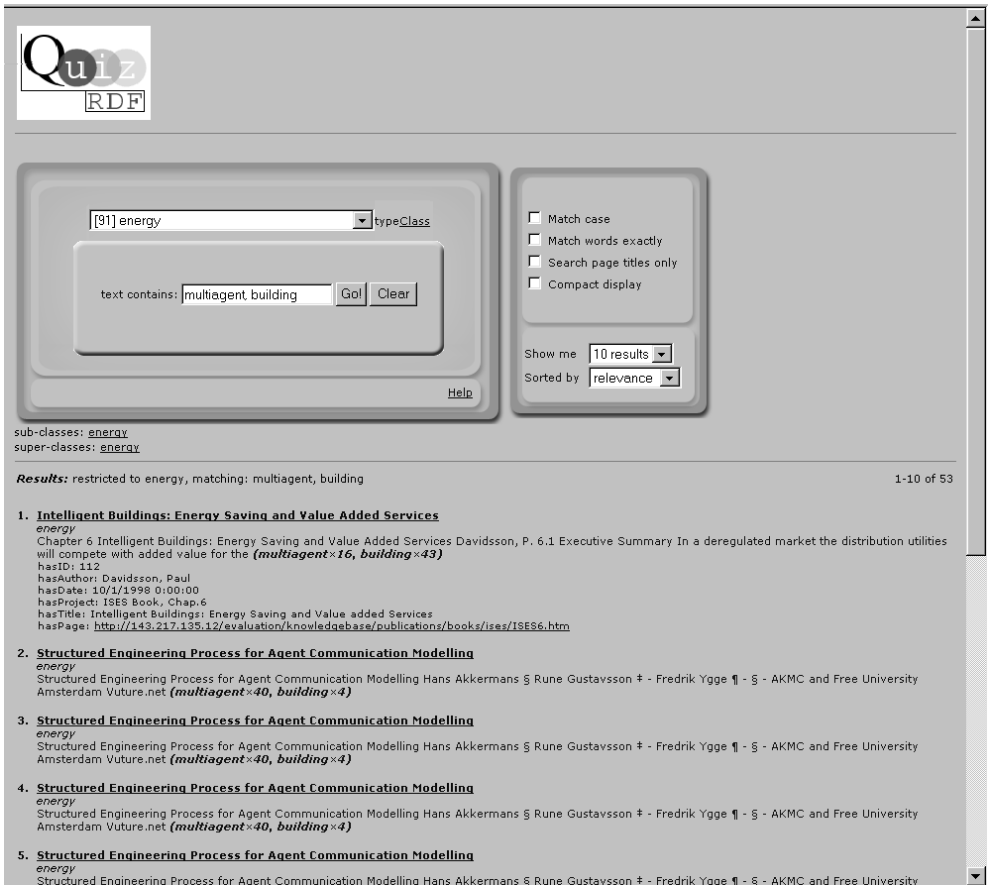

Figure 2.9: The query interface of QuizRDF

QuizRDF: Full Text Searching plus RDF Querying

QuizRDF[3] (Krohn, 2001) combines full text searching with RDF querying. This combined approach seems to be very promising due to the fact that RDF-annotated information resources are likely to be complemented by non-annotated information for a considerable period to come, and that any given RDF description of a set of resources will give one particular perspective on the information described. QuizRDF can be used like a conventional Internet search engine by entering a set of search terms or a natural language query and produces a list of links to relevant Web pages in the usual way.

However, QuizRDF's indexing and retrieval technique is also designed to use domain knowledge that is made available in the form of ontologies specified as OIL core. RDF resources are Web pages or (parts thereof) and such pages or segments are effectively ontological instances. Correspondingly, resource types are ontological classes. The information items processed by QuizRDF are RDF resources, which may be Web pages or parts thereof. During indexing QuizRDF assigns content descriptors to RDF resources. Content descriptors of a resource are terms (words and phrases) that QuizRDF obtains from a full text analysis of the resource content and from process-

[3]Due to legal matters the formally known RDFferret is now being called QuizRDF.

ing all literal values that are directly related by a property. They also retain structural information about the ontology.

In QuizRDF the user can select from a list of all the resource types stored in the index. When searching by selecting a resource type, QuizRDF adjusts its result list to show only resources of the selected type. The user is also presented with a search and navigation area. The search area shows the attributes of the selected resource type. For each attribute the user can input a search criterion. QuizRDF combines the search criteria entered and matches the resulting query against its ontology-based index.

In addition, resource types (ontological classes) related by some property to the currently selected type are displayed as hyperlinks. Clicking on such a type then selects that type and in turn displays those types that are related to it. Thus the user can browse the ontology in a natural and intuitive way.

Figure 2.9 shows a typical initial query by a user taken from the EnerSearch case study. The user has entered a free text query for information about "multiagent" and "building" and refined the query with a search for the class "energy" from an underlying ontology. The search engine has returned a ranked list of 53 documents containing the terms. When returning the result documents, QuizRDF has also compiled a list of the classes to which each document belongs. This class list is then made available to the user via the drop-down list referred to. The user can then refine the search results by selecting one of the classes from the list (like the here chosen "energy").

OntoShare: Community Support

OntoShare (Davies et al., 2002; Duke & Davies, 2001; Duke & van der Meer, 2002) enables the storage of best practice information according to an ontology and the automatic dissemination of new best practice information to relevant co-workers. It also allows users to browse or search the ontology in order to find the most relevant information to the problem that they are dealing with at any given time. The ontology helps new users to navigate and acts as a schema for storing key learning and best practices accumulated through experience. In addition, the ontology helps users to become familiar with new domains. It provides a sharable structure for the knowledge base, and a common language for communication between user groups. Each user can define his own relevant parts of the ontology (*i.e.* personal concepts) that are integrated into a single coherent ontology available to all users (*cf.* Figure 2.6, taken from the BT case study).

Spectacle: Information Presentation

Spectacle (Fluit et al., 2002) is a content presentation platform featuring custom-made information presentations, aimed at supporting the information needs of its users. This means not only that the right information should be delivered to the user, but also that it needs to be presented (structured, formatted, rendered) in a manner appropriate for that specific user.

Spectacle is used to disclose both the content of databases, document repositories and other enterprise information sources, as well as the semantics of that information from Semantic Web resources. The platform consists of the Spectacle server and programming libraries for generating both Web-based and graphical information presentations.

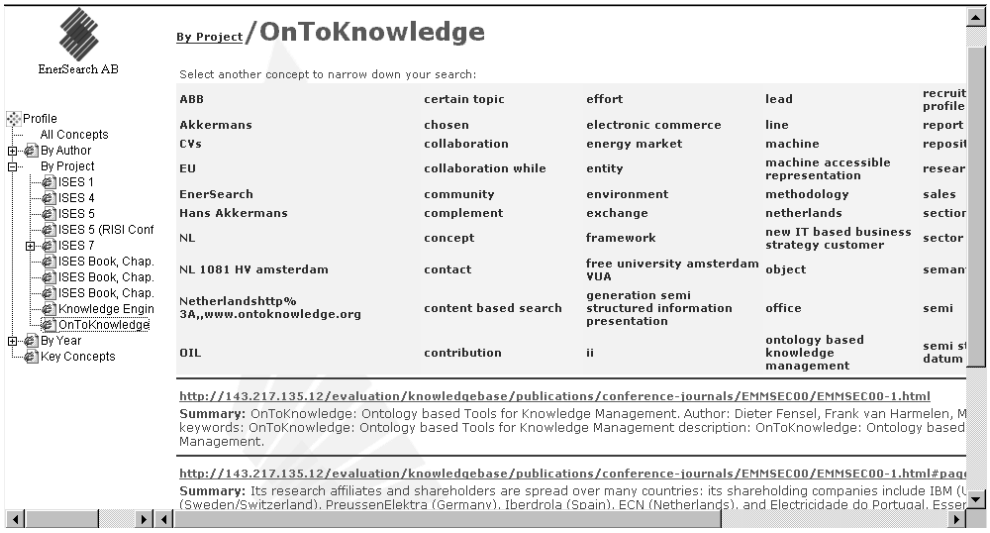

Figure 2.10: Provision of navigational structures with Spectacle

For the end user, Spectacle transforms the task of gathering information from a search task (formulating explicit queries) to a browsing task (using navigation heuristics) by presenting each user with the navigational means appropriate for his or her task. This results in more efficiency in retrieving the right information, both in terms of retrieval accuracy as well as time spent on the task.

Spectacle can present information in two different ways: (i) it can create *hypertext interfaces*, containing selected content, design and an appropriate navigation structure, based on the semantics of the information, (ii) it can present the information by *graphical visualization*.

A key benefit of the first approach is that it allows for an easy and flexible presentation of the same information in different ways, for each of the envisioned tasks or user groups. Furthermore, it has all the usual benefits of a generated Web site (like having a consistent design, being up-to-date) and it also takes advantage of the expressivity and flexibility provided by Semantic Web standards such as RDF, RDF Schema and DAML+OIL.

A benefit of the second approach is that it can offer insights and kinds of information access that are not possible with conventional publishing methods such as Web sites. For example, overview and analysis of large sets of objects requires an effective and compact graphical presentation. Similarly, presentation of the relations between these objects is virtually impossible without the support of a graphical visualization.

Figure 2.10, taken from the EnerSearch case study, shows an example for the first approach. On the left side the navigational view generated out of the underlying ontology with the selected concept “OnToKnowledge”, on the upper right side the current navigational path “By project/OnToKnowledge”, below other available concepts like “ABB” or “Akkermans” and, last but not least, on the lower left side the relevant set of documents for the selected navigational path.

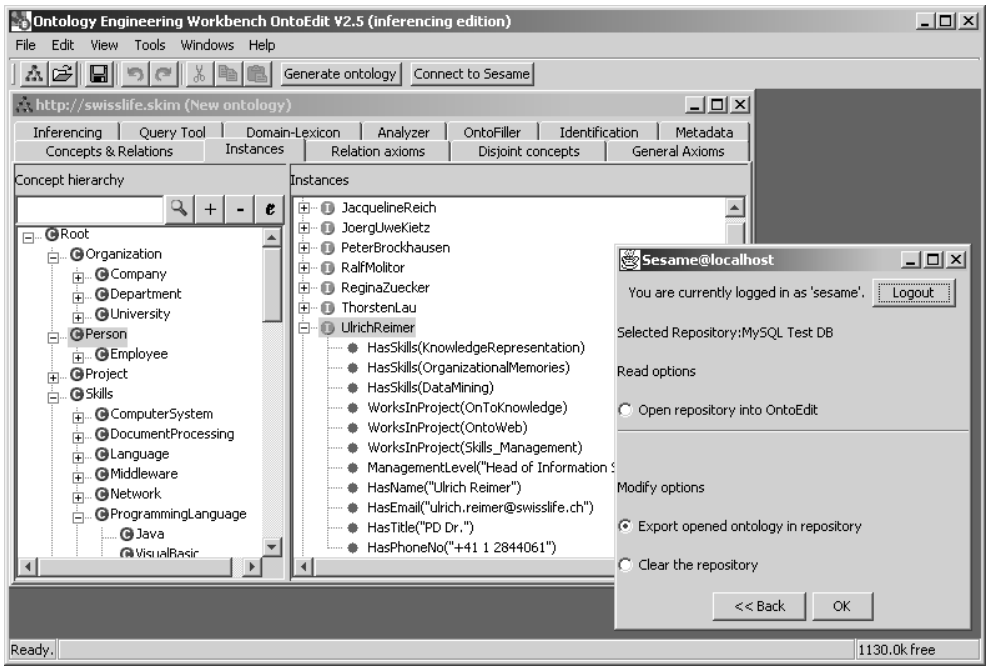

Figure 2.11: Ontology development with OntoEdit

OntoEdit: Ontology Development

OntoEdit (Sure et al., 2002a; 2002b; Sure & Studer, 2001a) is a collaborative ontology engineering environment that is easily expandable through a flexible plug-in framework (Handschuh, 2001). OntoEdit supports ontology engineers while inspecting, browsing, codifying and modifying ontologies in each step of the Knowledge Meta Process (*cf.* Chapter 3 or, *e.g.* (Staab et al., 2001)).

Modeling ontologies using OntoEdit involves modelling at a conceptual level, *viz.* (i) as independently of a concrete representation language as possible, and (ii) using GUI's representing views on conceptual structures (concepts, concept hierarchy, relations, axioms) rather than codifying conceptual structures in ASCII. In addition, OntoEdit provides a simple instance editor to insert facts according to a modelled ontology. The conceptual model of an ontology is stored internally using a powerful ontology model, which can be mapped onto different, concrete representation languages (*e.g.* OIL core, DAML+OIL or RDF(S)). Ontologies can be directly imported from and exported to Sesame.

The core functionalities of OntoEdit were expanded by several plugins to meet the requirements from the case studies – *e.g.* OntoKick and Mind2Onto (Sure et al., 2002a), SesameClient-Plugin, OntoFiller and, last but not least, OntoCleanPlugin (Sure et al., 2002b). They will be explained further in Chapter 3.

Figure 2.11 shows an ontology opened in OntoEdit taken from the Swiss Life case study. On the left side of the "ontology window" there is the concept is-a hierarchy with the chosen concept "Person", on the right side there is a list showing all instances for the selected concept, *e.g.* the selected instance "UlrichReimer". On the right side of the screenshot one sees an opened window of a connection to a Sesame repository. The user "sesame" is logged in at the Sesame repository

running on "localhost", currently the option "export opened ontology in repository" is chosen to upload the opened ontology to Sesame. Multiple ontologies can be opened at the same time and multiple connections to various Sesame repositories can be opened at the same time.

Ontology Middleware Module: Integration Platform

The Ontology Middleware Module[4] (OMM, *cf.* (Kiryakov et al., 2002b; 2002a)) can be seen as "administrative" software infrastructure that makes the knowledge management tools easier for integration in real-world applications. The major features supported are:

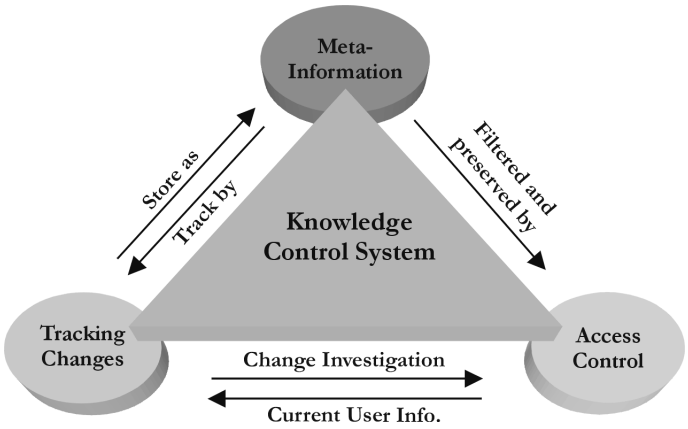

Figure 2.12: Features of the Knowledge Control System

- Change management for ontologies allows work with, revert to, extraction, and branching of different states and versions (*cf. e.g.* the next subsection on OntoView);
- Access control (security) system with support for role hierarchies including comprehensive and precise restrictions (down to object/record-level) that enable business-logic enforcement;
- Meta-information for ontologies, specific resources (classes, instances), and statements.

These three aspects are tightly integrated to provide the same level of the handling of knowledge in the process of its development and maintenance as source control systems (such as *e.g.* the Concurrent Versions System (CVS)[5]) provide for software. On the other hand, for end-user applications, OMM can be seen as equivalent to the database security, change tracking and auditing systems. Our OMM is carefully designed to support both use cases.

In a nutshell, OMM extends the storage and query facilities of Sesame with a Knowledge Control System (KCS, *cf.* Figure 2.12), additional support for multi-protocol access (*e.g.* HTTP, RMI, SOAP) and reasoning services.

[4]More information for OMM and BOR including an online demo can be found at (Ontology Middleware Module, 2002; BOR, 2002; Ontology Middleware Module Demo, 2002).

[5]http://www.cvshome.org/

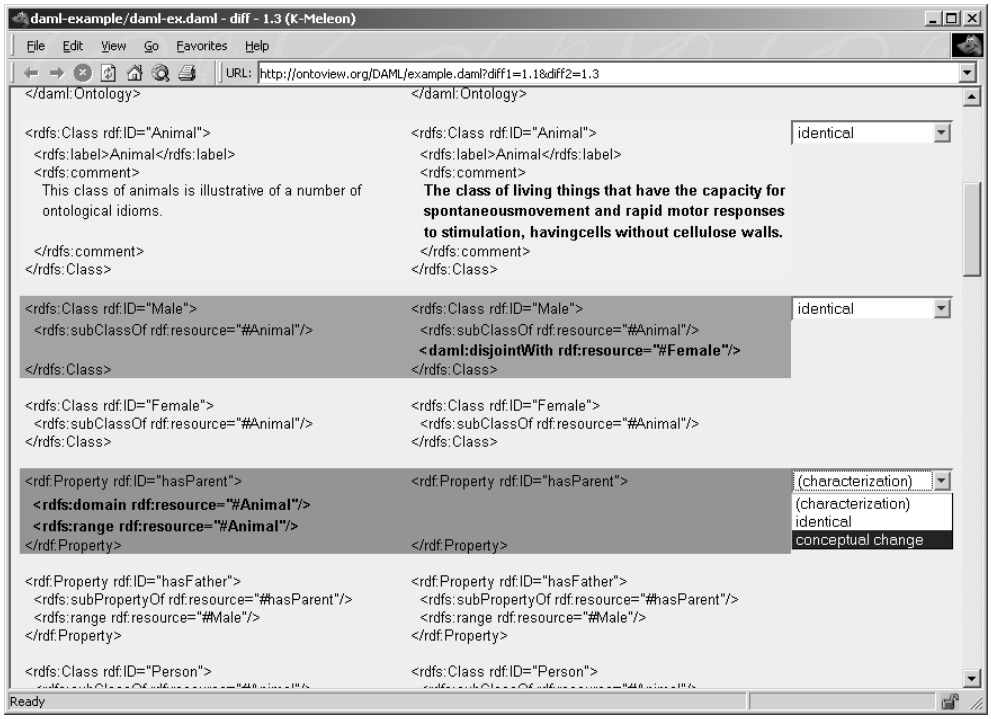

Figure 2.13: The result of a comparison of two ontologies with OntoView

An example for a reasoning service is BOR (Simov & Jordanov, 2002) – a reasoner that is currently being developed and complies with the DAML+OIL model-theoretic semantics. It is a modular component that can be plugged in to extend the query facilities already provided *e.g.* by Sesame. It addresses most of the classic reasoning tasks for description logics, including realization and retrieval. Few innovative services, such as model checking and minimal ontology extraction, are also integral part of the system. The full set of functional interfaces will allow a high level of management and querying of DAML+OIL ontologies.

OntoView: Change Management for Ontologies

OntoView (Klein & Fensel, 2002) is a change management tool for ontologies and is part of the Ontology Middleware Module (it is not separately shown in Figure 2.8). Change management is especially important when ontologies will be used in a decentralized and uncontrolled environment like the Web, where changes occur without co-ordination. The main function of OntoView is to provide a transparent interface to arbitrary versions of ontologies. To achieve this, it maintains an internal specification of the relation between the different variants of ontologies. This specification consists of three aspects: (i) the *meta-data* about changes (author, date, time etc), (ii) the *conceptual relations* between versions of definitions in the ontologies, and (iii) the *transformations* between them. This specification is partly derived from the versions of ontologies themselves, but also uses additional human input about the meta-data and the conceptual effects of changes.

To help the user to specify this information, OntoView provides the utility to compare versions of ontologies and highlight the differences. This helps in finding changes in ontologies, even if

those have occurred in an uncontrolled way, *i.e.* possibly by different people in an unknown order. The comparison function is inspired by UNIX `diff`, but the implementation is quite different. Standard `diff` compares file version at line-level, highlighting the lines that textually differ in two versions. OntoView, in contrast, compares version of ontologies at a *structural* level, showing which definitions of classes or properties are changed.

There are different types of change. Each type is highlighted in a different color, and the actually changed lines are printed in boldface. An example of the visual representation of the result of a comparison is shown in Figure 2.13.

The comparison function distinguishes between the following types of change:

- Non-logical change, *e.g.* in a natural language description. This are changes in the label of a concept or property, or in the comment inside definitions.
- Logical definition change. This is a change in the definition of a concept that affects its formal semantics. Examples of such changes are alterations of subclass statements, or changes in the domain or range of properties. Additions or deletions of local property restriction in a class are also logical changes. The second and third change in the Figure 2.13 (class "Male" and property "hasParent") are examples of such changes.
- Identifier change. This is the case when a concept or property is given a new identifier, *i.e.* a renaming.
- Addition of definitions.
- Deletion of definitions.

The comparison function also allows the user to specify the conceptual implication of the changes. For the first three types of changes, the user is given the option to label them either as "identical" (*i.e.* although the specification is changing, it still refers to the same concept), or as "conceptual change". In the latter case, the user can specify the conceptual relation between the two version of the concept. For example, by stating that the property "$\text{hasParent}_{1.0}$" is a subproperty of "$\text{hasParent}_{2.0}$".

Another function is the possibility to analysis effects of changes. Changes in ontologies do not only affect the data and applications that use them, but they can also have unintended, unexpected and unforeseeable consequences in the ontology itself (McGuinness et al., 2000). The system provides some basic support for the analysis of these effects. First, on request it can also highlight the places in the ontology where conceptually changed concepts or properties are used. For example, if a property "hasChild" is changed, it will highlight the definition of the class "Mother", which uses the property "hasChild". This function can also exploit the transitivity of properties to show the propagation of possible changes through the ontology. A foreseen second effect analysis feature is a connection to FaCT (Horrocks, 1998), which allows to check the formal consistency of the suggested conceptual relations between different versions of definitions.

Sesame: Repository for Ontologies and Data

Sesame[6] (Jeen Broekstra, 2002; Broekstra & Kampman, 2001) is a system that allows persistent storage of RDF data and schema information and subsequent online querying of that information. Sesame has been designed with scalability, portability and extensibility in mind.

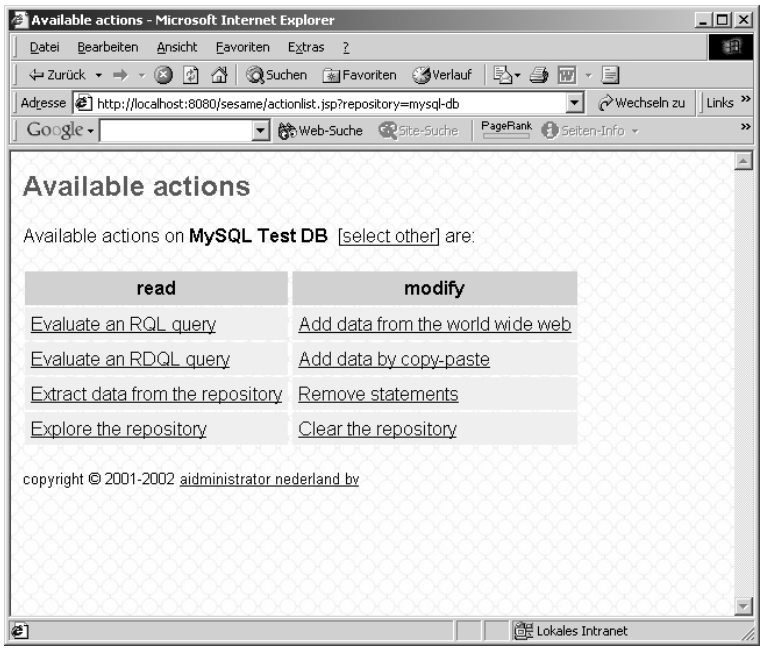

Figure 2.14: Sesame: repository for ontologies and data

Sesame itself has been implemented in Java, which makes it portable to almost any platform. It also abstracts from the actual repository used by means of a standardized API. This API makes Sesame portable to any repository (DBMS or otherwise) that is able to store RDF triples. Currently, only implementations based on DBMS's exist. At the same time, this API enables swift addition of new modules that operate on RDF and RDF Schema data.

One of the most prominent modules of Sesame is its query engine. This query engine supports an OQL-style query language called RQL (*cf.* (Karvounarakis et al., 2001; Broekstra et al., 2000; Broekstra & Kampman, 2000)). RQL supports querying of both RDF data (*e.g.* instances) and schema information (*e.g.* class hierarchies, domains and ranges of properties). RQL also supports path-expressions through RDF graphs, and can combine data and schema information in one query. The streaming approach used in Sesame (data is processed as soon as available) makes for a minimal memory footprint. This streaming approach also makes it possible for Sesame to scale to huge amounts of data. In short, Sesame can scale from devices as small as palm-top computers to powerful enterprise servers.

A final feature of Sesame is its flexibility in communicating with other tools. Currently, Sesame itself only supports communication over HTTP, support for other protocols is added through the Ontology Middleware Module on top of it. Sesame has now been released as Open

[6]More information including an online demo can be found at (Sesame, 2002), the Source Forge project website can be found at (Sesame Source Forge Project, 2002)

Source under the GNU Lesser General Public License (LGPL).

Figure 2.14 shows available actions at the web-interface for a Sesame repository running on "localhost". Currently the database "MySQL Test DB" of this Sesame repository is chosen and the user has several options for reading or modifying the content of this database.

CORPORUM: Information Extraction

The CORPORUM toolset (Engels & Bremdal, 2001a; 2000; 2001b; 2002) consists of two parts, *viz.* OntoExtract (*cf.* Figure 2.15) and OntoWrapper (*cf.* Figure 2.16), and has two related, though different, tasks: interpretation of natural language texts and extraction of specific information from free text.

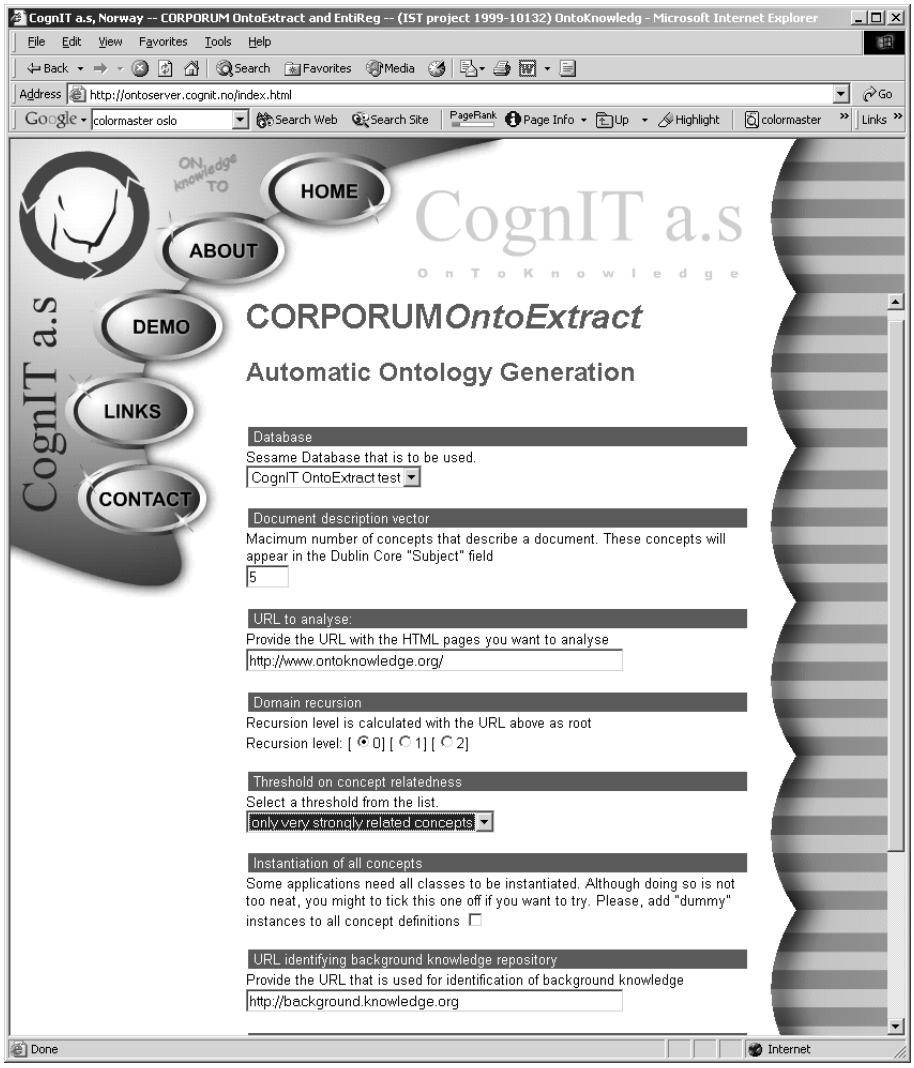

Figure 2.15: OntoExtract: Automatic ontology generation

Whereas the former process can be performed autonomously by CORPORUM tools, the latter

task requires a user who defines business rules for extracting information from tables, (phone) directories, home-pages, etc. Although this task is not without its challenges, most effort focuses on the former task, which involves natural language interpretation on a syntactic and lexical level, as well as interpretation of the results of that level (discourse analysis, co-reference and collocation analysis, etc.).

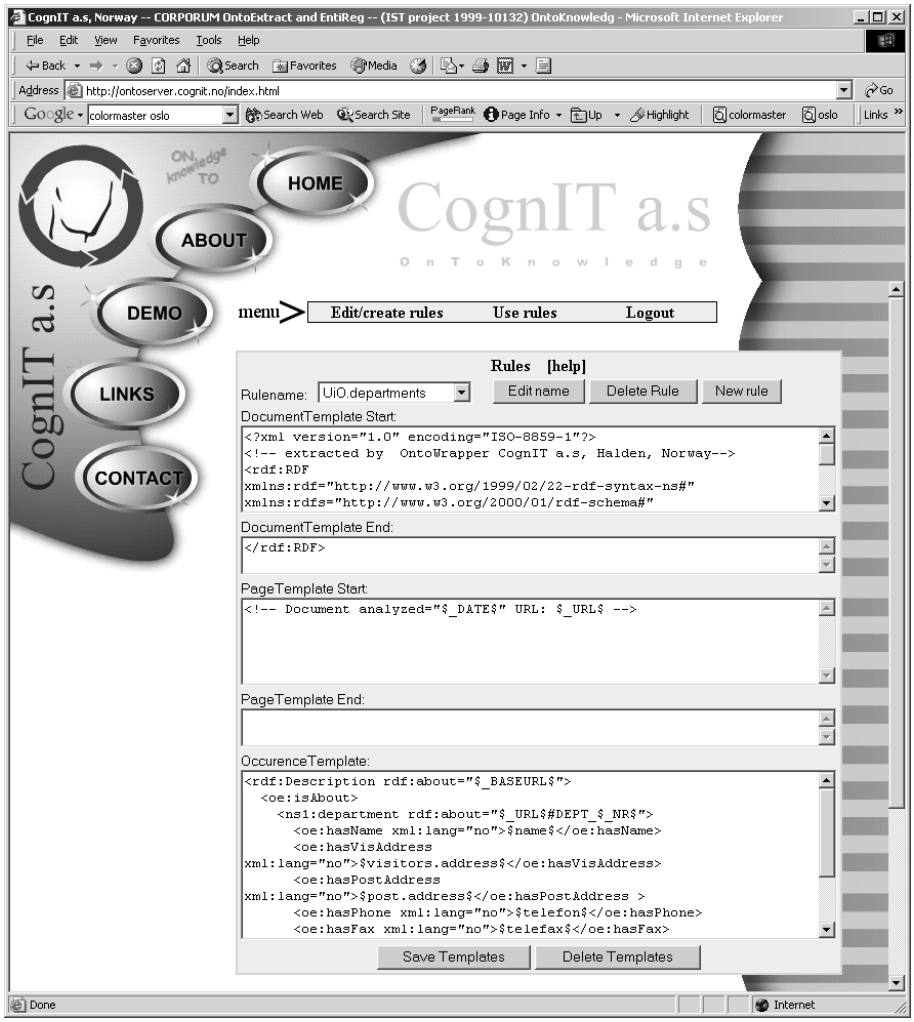

Figure 2.16: OntoWrapper: Information extraction

The CORPORUM system outputs a variety of (symbolic) knowledge representations, including semantic (network) structures and visualizations thereof, light-weight ontologies, text summaries, automatically generated thesauri (related words/concepts), etc. Thus, extracted information is represented in RDF(S)/DAML+OIL, augmented with Dublin Core Meta Data wherever possible, and submitted to the Sesame data repository mentioned previously.

Typically, the CORPORUM system does not incorporate background knowledge itself, but relies on its extraction and analysis capabilities in combination with any knowledge available in the Sesame repository. The availability of knowledge, however, is not a prerequisite.

2.5.2 OIL: Inference Layer for the Semantic World Wide Web

The OTK tool suite discussed above exploits ontologies as its common operating ground: *e.g.* an ontology was created and refined manually (OntoEdit) or extracted semi-automatically (OntoExtract), raw information sources were structured on the basis of an ontology (OntoWrapper), this structured data was stored and managed in an ontology-based repository (Sesame and OMM), and the data could be queried using the vocabulary from an ontology. Finally, information could be shared (OntoShare), searched (QuizRDF) or browsed (Spectacle) by users on the basis of such ontological vocabularies.

All of this of course requires the existence of a language to express such ontologies. Some basic requirements for such a language are:

- sufficient expressivity for the applications and tasks mentioned in the preceding sections
- sufficiently formalized to allow machine processing
- integrated with existing Web technologies and standards

Although much work has been done on ontology languages in the AI community (see *e.g.* (Corcho & Gomez Perez, 2000) for a recent overview), it is particularly the 3rd requirement that motivated us to design a new language (baptized OIL) for our purposes. In this section, we will briefly describe the constructions in the OIL language, and then discuss its most important features and design decisions.

Combining Description Logics with Frame Languages

The OIL language (Fensel et al., 2000b; 1999; 2000c; Fensel, 2002) is designed to combine frame-like modelling primitives with the increased (in some respects) expressive power, formal rigor and automated reasoning services of an expressive description logic. OIL also comes "Web enabled" by having both XML and RDFS based serializations (as well as a formally specified "human readable" form, which we will use here[7]). The frame structure of OIL is based on XOL (Karp et al., 1999), an XML serialization of the OKBC-lite knowledge model (Chaudhri et al., 1998). In these languages classes (concepts) are described by frames, which consist of a list of super-classes and a list of slot-filler pairs. A slot corresponds to a role in a DL, and a slot-filler pair corresponds to either a universal value restriction or an existential quantification. OIL extends this basic frame syntax so that it can capture the full power of an expressive description logic. These extensions include:

- Arbitrary boolean combinations of classes (called class expressions) can be formed, and used anywhere that a class name can be used. In particular, class expressions can be used as slot fillers, whereas in typical frame languages slot fillers are restricted to being class (or individual) names.

[7] http://www.ontoknowledge.org/oil/syntax/

- A slot-filler pair (called a slot constraint) can itself be treated as a class: it can be used anywhere that a class name can be used, and can be combined with other classes in class expressions.
- Class definitions (frames) have an (optional) additional field that specifies whether the class definition is primitive (a subsumption axiom) or non-primitive (an equivalence axiom). If omitted, this defaults to primitive.
- Different types of slot constraint are provided, specifying universal value restrictions, existential quantification and various kinds of cardinality constraint.
- Global slot definitions are extended to allow the specification of superslots (subsuming slots) and of properties such as transitivity, and symmetry.
- Unlike many frame languages, there is no restriction on the ordering of class and slot definitions, so classes and slots can be used before they are defined.
- OIL also provides axioms for asserting disjointness, equivalence and coverings with respect to class expressions.

Many of these points are standard for a description logic, but are novel for a frame language. OIL is also more restrictive than typical frame languages in some respects. In particular, it does not support collection types other than sets (*e.g.* lists or bags), and it does not support the specification of default fillers. These restrictions are necessary in order to maintain the formal properties of the language (*e.g.* monotonicity) and the correspondence with description logics.

Web Interface

As part of the Semantic Web activity of the W3C, a very simple Web-based ontology language had already been defined, namely RDF Schema. This language only provides facilities to define class- and property-names, inclusion axioms for both classes and properties (subclasses and subproperties), and to define domain and range constraints on properties. Instances of such classes and properties are defined in RDF.

OIL has been designed to be a superset of the constructions in RDF Schema: all valid RDF Schema expressions are also valid OIL expressions. Furthermore, the syntax of OIL has been designed such that any valid OIL document is also a valid RDF(S) document when all the elements from the OIL-namespace are ignored. The RDF Schema interpretation of the resulting subdocument is guaranteed to be sound (but of course incomplete) with respect to the interpretation of the full OIL document.

This guarantees that any RDF Schema agent can correctly process arbitrary OIL documents, and still correctly capture some of the intended meaning. The full details of how this has been achieved, and the trade-offs involved in this can be found in (Broekstra et al., 2001).

Layering

For many of the case study applications from section 2.4, it is unlikely that a single language will be ideally suited for all uses and all users. In order to allow users to choose the expressive

power appropriate to their application, and to allow for future extensions, a layered family of OIL languages has been described. The sublanguage OIL Core has been defined to be exactly the part of OIL that coincides with RDF(S). This amounts to full RDF(S), without some of RDF's more dubious constructions: containers and reification.

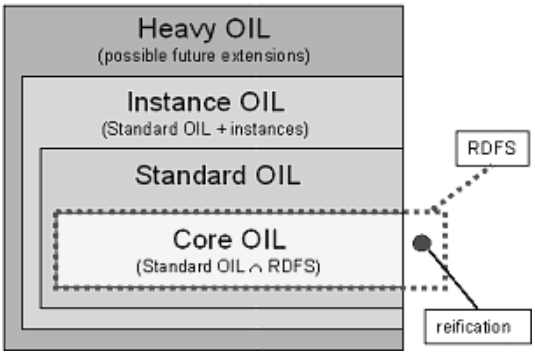

Figure 2.17: The layered language model of OIL

The standard language, is called "Standard OIL", and when extended with the ability to assert that individuals and tuples are, respectively, instances of classes and slots), is called "Instance OIL". Finally, "Heavy OIL" is the name given to a further layer that will include as yet unspecified language extensions. This layering is depicted in Figure 2.17.

Figure 2.18 illustrates an OIL ontology (using the human readable serialization), developed in a skills management case study by Swiss Life.

```
class-def Department
instance-of ITDept Department
class-def Skills
    slot-constraint SkillsLevel cardinality 1
slot-def HasSkills
    domain Employee
    range Skills
slot-def WorksInProject
    domain Employee
    range Project
    inverse ProjectMembers
class-def defined ITProject
    subclass-of Project
    slot-constraint ResponsibleDept has-value ITDept
slot-def ManagementLevel
    domain Employee
    range one-of "member" "head-of-group"
                 "head-of-dept" "CEO"
class-def Publishing
    subclass-of Skills
class-def DocumentProcessing
    subclass-of Skills
class-def DesktopPublishing
    subclass-of Publishing and DocumentProcessing
instance-of GeorgeMiller Employee
related HasSkills GeorgeMiller DesktopPublishing3
instance-of DesktopPublishing3 DesktopPublishing
related SkillsLevel DesktopPublishing3 3
```

Figure 2.18: OIL illustration

The following points are noteworthy:

- `Skills` are restricted to being of a single level trough a cardinality constraint
- `WorksInProject` and `ProjectMembers` are defined to be each others inverse
- `ITProjects` are defined to be exactly those projects whose `ResponsibleDept` is the `ITDept`
- `DeskTopPublishing` is defined to be in the intersection of `Publishing` and `DocumentProcessing`

Current Status

Meanwhile, OIL has been adopted by a joined EU/US initiative that developed a language called DAML+OIL[8], which has now been submitted to the Ontology Working Group of the W3C[9], the standardization committee of the WWW. We can soon expect a recommendation for a Web ontology language; hopefully, it will feature many of the elements and aspects on which OIL is based.

Future Developments

In November 2001, the W3C started a Working Group (WG) for defining a Web Ontology language. This WG is chartered to take DAML+OIL as its starting point, now continuing on the evolving standard OWL (Ontology Web Language). Over 40 of the W3C members from academia and industry are currently participating in this effort. One of your core recommendations for this working group that we distilled from our own experiences is the urgent need for a layering of such languages.

Other efforts are underway to define extensions for the ontology language, such as an ontology-query language, or an extension with rules (which would allow for example role chaining, as done in Horn logic).

2.6 Usage of Tools in Case Studies

Similar to LEGO pieces the OTK tool set can be plugged together in various ways to meet specific requirements. Each case study had different settings that are briefly shown in the following subsections.

It is noteworthy that BOR was not explored in any case study due to the fact that its developing partner OntoText was introduced to the project at a late stage. The tool was designed and

[8] For information about DAML+OIL *cf.* http://www.daml.org/2001/03/daml+oil-index and http://www.w3.org/TR/daml+oil-reference.

[9] *cf.* http://www.w3.org/2001/sw/WebOnt/ for information about the Ontology Working Group, *cf.* http://www.w3.org/TR/webont-req/ for the latest W3C working draft on requirements for a Ontology Web Language

implemented in an impressively short period of time, but was finished in the final phase of On-To-Knowledge.

2.6.1 Swiss Life

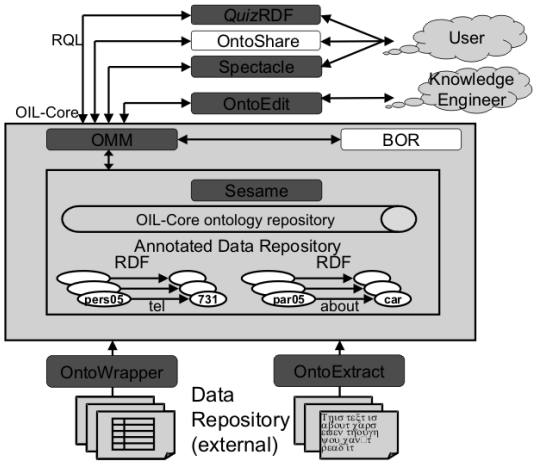

Figure 2.19: Covering of tools @ Swiss Life case study

The case study at Swiss Life was the first one to start, unfortunately it was also the first one to end. An internal restructuring at Swiss Life led to a pre-final closure of the case study. However, it was planned to cover a broad range of tools in the two parts (*cf.* Figure 2.19, the dark grey shaded tools are partially used and were partially planned to be used in this case study).

2.6.2 BT

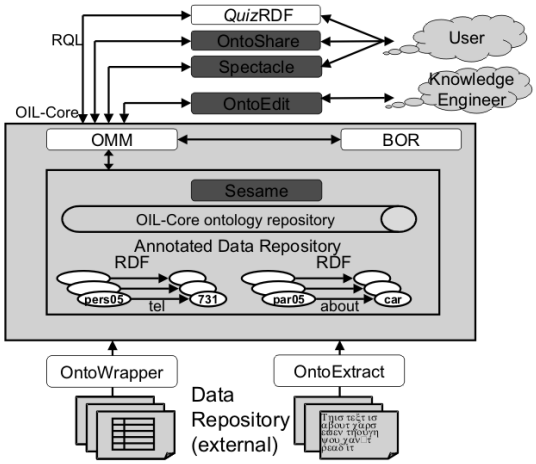

Figure 2.20: Covering of tools @ BT case study

The BT case study is circled around OntoShare (*cf.* Figure 2.20, the dark grey shaded tools are used in this case study). OntoEdit is used to develop the underlying ontology, Sesame serves

as central storage and query facility (e.g. the user profiles can be expressed as RQL queries). Though not included into the evaluation, we used the information presentation layer of Spectacle as a web-based access to the OntoShare store.

2.6.3 EnerSearch

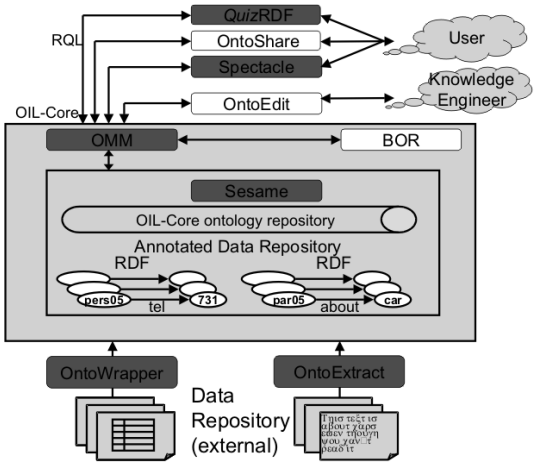

Figure 2.21: Covering of tools @ EnerSearch case study

Main parts of the EnerSearch case study consist of user-focused and technology-focused evaluation of OTK tools, i.e. (i) in a field experiment QuizRDF and Spectacle were compared against the EnerSEARCHer and (ii) Sesame and OMM were tested on scalability and interoperability issues. OntoExtract and OntoWrapper provided the necessary support for generating automatically ontologies for the user-focussed scenario.

2.7 OTK Methodology

The Institute AIFB (Germany) provides a methodology that includes guidelines for introducing ontology-based knowledge management concepts and tools into enterprises, helping knowledge providers and seekers to present knowledge efficiently and effectively.

Within the project the methodology provider is intermediating between the technology providers and the case study providers and captures lessons learned from the OTK case studies while applying the OTK tool set. This document is the final version of the OTK methodology workpackage (see (Staab et al., 2001; Schnurr et al., 2000; Sure & Studer, 2001b; 2002) the previous versions).

We will now present our methodology for introducing and maintaining ontology based knowledge management solutions into enterprises.

Chapter 3

Knowledge Meta Process

In this chapter we start by discussing the relevant processes for setting up knowledge management (KM) applications, *i.e.* "Knowledge Meta Process", "Human Issues" and "Software Engineering". The focus in this work is on the Knowledge Meta Process.

We illustrate in the following sections each of the steps (phases) of the Knowledge Meta Process (the "light grey circle" in Figure 2.1), *i.e.* the "Feasibility Study", "Kickoff", "Refinement", "Evaluation" and "Application & Evolution". For each phase we illustrate (i) relevant sub-steps in detail, (ii) the outcome, (iii) decisions to be taken at the end of each phase, (iv) tool support from OTK tools and (v) experiences from applying the tools in the case studies according to the methodology.

We will show some selected major influences from the other mentioned processes, i.e. "Human Issues" and "Software Engineering".

While the Knowledge Meta Process guides and supports the *initial set up* of an ontology based application, the following Chapter 4 subsequently depicts the cyclic process of *using* an application, *i.e.* the Knowledge Process.

3.1 Implementation and Invention of KM Applications

To implement and invent any KM application, one has to consider different processes (*cf.* Figure 3.1). We experienced mainly three major process that influenced the project, *i.e.* "Knowledge Meta Process", "Human Resource Management" and "Software Engineering". These processes are not complete separate but also interfere. As mentioned in Section 2.1, KM is an inherently interdisciplinary subject which is not only governed by information technology (IT). Hence, one needs to keep the balance between human problem solving and automated IT solutions. As a rule of thumb it was carefully estimated by KM experts at a "Dagstuhl Seminar on Knowledge Management"[1] that in "real life" IT support cannot cover more than 10—30% of KM (*cf.* (O'Leary & Studer, 2001)).

Human issues (HI) and the related cultural environment of organizations heavily influence the acceptance of KM. It is often mentioned in discussions that the success of KM – and especially

[1] http://dagstuhl-km-2000.aifb.uni-karlsruhe.de/

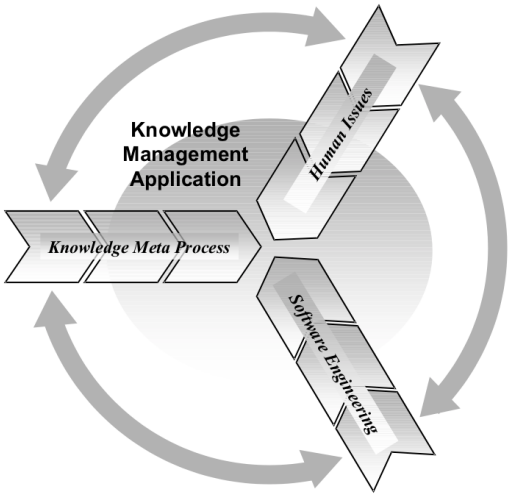

Figure 3.1: Relevant processes for implementing and inventing KM applications

KM applications – strongly depends on the acceptance by the involved people. As a consequence, "quick wins" are recommended for the initial phase of implementing any KM strategy. The aim is to quickly convince people that KM is useful for them and adds value to their daily work.

Software engineering (SE) guides (or, at least, should guide!) the design and implementation of any software – including KM applications. There exist numerous well-known strategies to develop software, typically it is up to the developing team to pick one.

In the following sections we will now focus on the Knowledge Meta Process as the core process for On-To-Knowledge and illustrate some cross-links to the other mentioned processes.

3.2 Steps of the Knowledge Meta Process

The Knowledge Meta Process (*cf.* Figure 3.2) consists of five main steps. Each step has numerous sub-steps, requires a main decision to be taken at the end and results in a specific outcome:

- The main stream indicates steps (phases), that finally lead to an ontology based KM application. The phases are: "Feasibility Study", "Kickoff", "Refinement", "Evaluation" and "Application & Evolution".
- Below every phase the most important sub-steps are sketched, *e.g.* "Refinement" consists of the sub-steps "Extract knowledge" and "Formalize" etc..
- Each document-flag above a phase indicates major outcomes of the step, *e.g.* "Kickoff" results in a "Semi-formal ontology description" etc..
- Each node above a flag represents the major decisions that have to be taken at the end to proceed to the next phase. The major outcomes typically serve as decision support for the decisions to be taken.

- The phases “Application & Evolution – Refinement – Evaluation – Application & Evolution” and “Evaluation – Refinement – Evaluation” may need to be performed in iterative cycles.

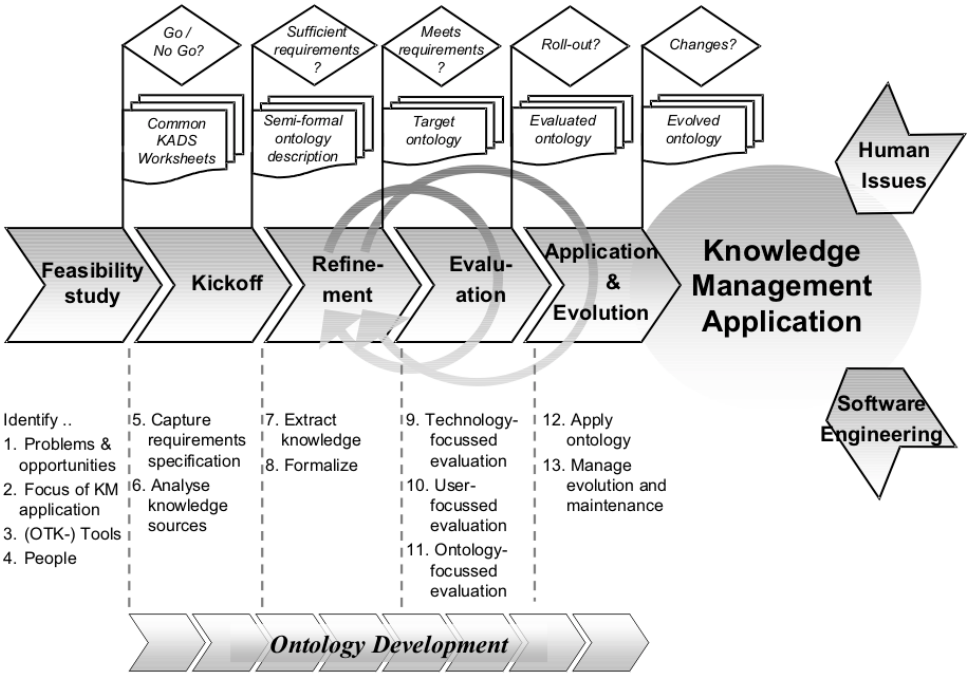

Figure 3.2: Knowledge Meta Process

3.3 Feasibility Study

Any knowledge management system can function satisfactorily only if it is properly integrated in the organization in which it is operational. Many factors other than technology determine success or failure of such a system (*cf.* Section 3.1). To analyze these factors, we initially have to proceed a *feasibility study*. *I.e.* we identify problem/opportunity areas and potential solutions, and put them into a wider organizational perspective. In general, a feasibility study serves as a decision support for economical, technical and project feasibility, in order to select the most promising focus area and target solution.

We focus on aspects of the feasibility study that help to identify (i) stakeholders related to a project divided into *users of the system* and *supporters of the system*, (ii) use cases describing the usage scenarios which we call *user driven use cases* and (iii) use cases supporting these user driven use cases which we call *supporting use cases*.

The well-known CommonKADS methodology (Schreiber et al., 1999) offers three models for performing feasibility studies: the organization, task, and agent model. The process of building these models proceeds in the following steps:

- Carry out a scoping and problem analysis study, consisting of two parts:
  - Identifying problem/opportunity areas and potential solutions, and putting them into a wider organizational perspective.
  - Deciding about economic, technical and project feasibility, in order to select the most promising focus area and target solution.
- Carry out an impacts and improvements study, for the selected target solution, again consisting of two parts:
  - Gathering insights into the interrelationships between the business task, actors involved, and use of knowledge for successful performance, and what improvements may be achieved here.
  - Deciding about organizational measures and task changes, in order to ensure organizational acceptance and integration of a knowledge system solution.

An overview of the process of organizational context modeling is given in Figure 3.3. Building the task, organization and agent model is done by following a series of steps supported by practical and easy-to-use worksheets and checklists (we refer for a detailed description of these steps to the CommonKADS methodology (Schreiber et al., 1999)).

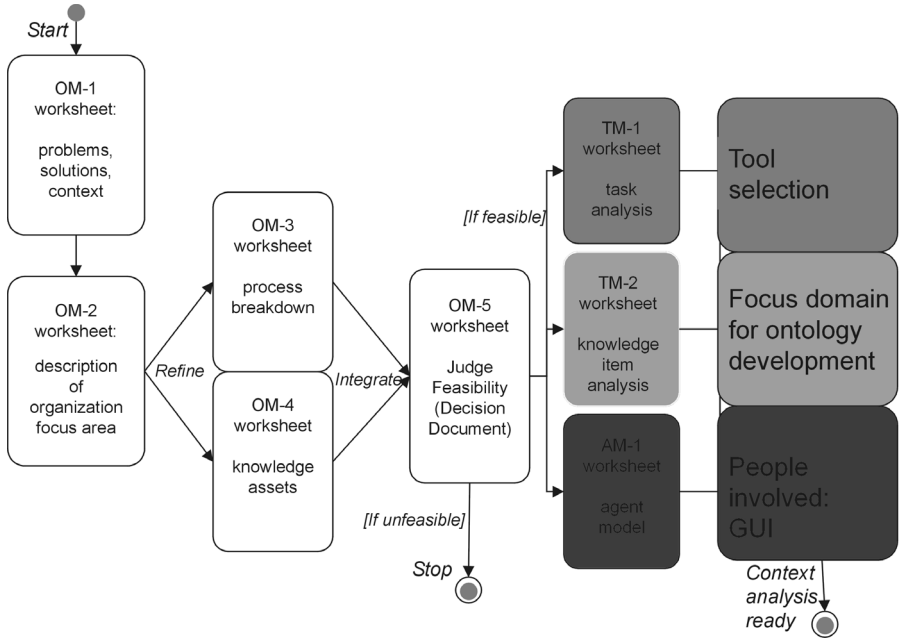

Figure 3.3: Modified CommonKADS steps

In the On-To-Knowledge scenario, the focus while performing a feasibility study lies on objectives and tools from the project itself (*cf.* (Fensel & van Harmelen, 2000)). *E.g.* the decision to perform specific case studies is already made as part of the project proposal.

As a next step according to CommonKADS one has to identify relevant tasks, agents carrying them out and knowledge items used by the agents while performing tasks[2].

3.3.1 Outcome

For the OTK purpose, the steps (*viz.* TM-1 worksheet – task analysis, TM-2 worksheet – knowledge item analysis and AM-1 worksheet – agent model) lead to a modified result as indicated in the dark shading in Figure 3.3.

3.3.2 Decision

Given that a "GO" decision finalizes the feasibility study, the results as described above serve as input for the kick off phase of the ontology development. Obviously, a "No GO" decision might lead to a complete stop or at least a modification of the project.

3.3.3 Experiences

Experiences in the OTK case studies showed, that the Meta Knowledge Process potentially includes further cycles: During every step it might come true that the project has to be modified or even is stopped at all. Typically a modification results in a new feasibility study, starting the Knowledge Meta Process from the very beginning.

In On-To-Knowledge we experienced both possibilities: (i) the pre-final closure of the Swiss Life case study and (ii) the complete reorganization of the BT case study, both due to internal restructurings at the companies.

3.4 Kickoff

3.4.1 Requirement Specification

In the kickoff phase the actual development of the ontology begins. Similar to requirements engineering and as proposed by (Lopez et al., 1999) we start with an ontology requirements specification document (ORSD). The ORSD describes what an ontology should support, sketching the planned area of the ontology application and listing, *e.g.*, valuable knowledge sources for the gathering of the semi-formal description of the ontology. The ORSD should guide an ontology engineer to decide about inclusion and exclusion of concepts and relations and the hierarchical structure of the ontology. In this early stage one should look for already developed and potentially reusable ontologies. In detail, the ORSD contains the following information (see also an example of an ontology requirements specification document in Figure 3.4 taken from the Swiss Life IAS case study).

[2]Some definitions according to CommonKADS: "A task is a *piece of work that needs to be done by an agent*." "An agent is any *human or software system able to execute a task* in a certain domain."

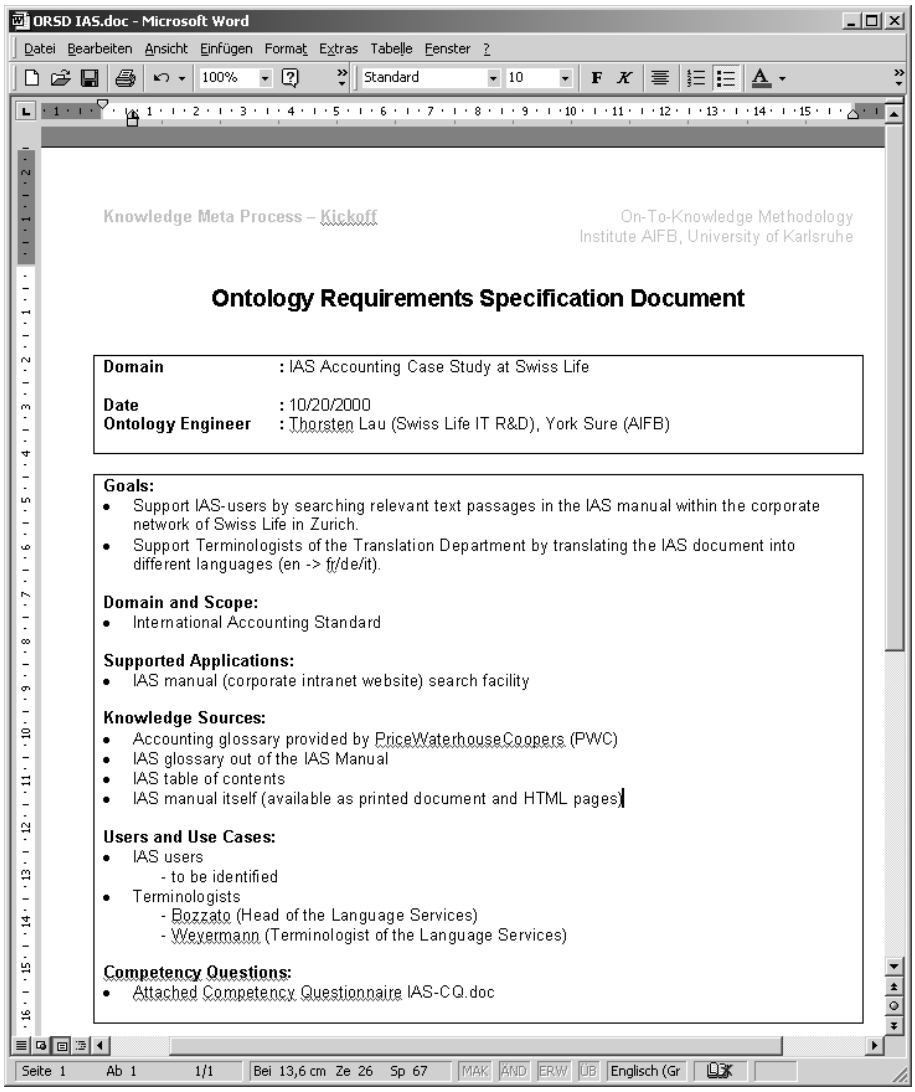

Figure 3.4: Ontology requirements specification document (ORSD)

Goal, Domain and Scope of the ontology

In the beginning one should specify the particular domain in use, which might help to identify already existing ontologies. The feasibility study made clear proposals about interesting areas to be supported by a knowledge management project. The ontology engineer may use the outcomes of the task analysis to describe the goal of the ontology. The following list gives some examples: ”The ontology serves as a means to structure skills and job profiles“, ”The ontology serves as a guideline for the knowledge distribution between the Human Resource department and the Research and Development department“, ”The ontology serves as a base for semantic search“.

Design Guidelines

Due to the nature of our case studies we focus on pragmatic design guidelines that help users who are not familiar with modeling. They might *e.g.* contain an estimation of the number of concepts and the level of granularity of the planned model. This estimation is based on the knowledge item analysis, a further outcome of the feasibility study. *E.g.* if the requirements analysis specified that an ontology should support browsing through a domain which includes around 100 concepts and the ontology engineer ended up with modeling 1000 concepts, either the ontology grew too big and should be modified to fulfill the requirements or the requirement specification is not up to date any longer and should be updated. Also one might specify common rules how to name concepts. A typical approach for a naming convention is to begin all concepts with capitals and all relations with small caps (see the Section 3.4.5 for our naming approach for concepts in OTK). Whatever rules one might specify, they should be used consistently when modeling an ontology. Ideally an ontology engineering tool should support to set these kinds of constraints and check it during the modeling process.

(Noy & McGuinness, 2001) proposes pragmatic guidelines for modeling ontologies. Especially domain experts not familiar with modeling (*e.g.* at Swiss Life) found these guidelines quite helpful. The guidelines are based on experiences with Protege that is an ontology editor similar to OntoEdit. Therefore the guidelines were easily applicable when using OntoEdit for modeling. We now give a brief overview of the guidelines.

- The ontology should not contain all the possible information about the domain: you do not need to specialize (or generalize) more than you need for your application (at most one extra level each way).
- The ontology should not contain all the possible relations of and distinctions among concepts in a hierarchy.
- Subconcepts of a concept usually (i) have additional relations that the superconcept does not have, or (ii) restrictions different from those of the superconcept, or (iii) participate in different relationships than the superconcepts. In other words, we introduce a new concept in the hierarchy usually only when there is something that we can say about this concept that we cannot say about the superconcept. As an exception, concepts in terminological hierarchies do not have to introduce new relations (this also holds for most of the "leightweight" ontologies developed in the case studies).
- If a distinction is important in the domain and we think of the objects with different values for the distinction as different kinds of objects, then we should create a new concept for the distinction.
- A concept to which an individual instance belongs should not change often.

More guidelines include *e.g.* naming conventions for concepts.

Knowledge Sources

The knowledge item analysis from the feasibility study serves as an important knowledge source at hand. The ontology engineer may here interview people and analyze documents to complete the list of knowledge sources for the domain in use. The following shows a partial list of knowledge sources as an example:

- domain experts (interviews, competency questionnaires)
- (reusable) ontologies
- dictionaries
- thesauri
- internal sources
  - databases
  - index lists
  - regulations
  - standard templates
  - product and project descriptions
  - technology white papers
  - telephone indices
  - web pages / site statistics
  - organization charts
  - employee role descriptions
  - business plans
- external documents

The usage of potentially reusable ontologies may improve the speed and quality of the development during the whole process. These ontologies might *e.g.* give useful hints for modeling decisions. If the available ontologies fulfill the requirements (*viz.* in the ORSD) one might even reuse an already existing ontology — or reuse it with slight modifications.

An ontology engineer should use all available knowledge sources based on their availability and reliability.

Some of these knowledge sources like *e.g.* databases might be directly integrated within the envisaged application. A key benefit of ontology based systems is the integrated access to heterogeneous and distributed knowledge sources. Examples include so-called “Semantic Portals” (Maedche et al., 2002). Especially the Swiss Life case study explored the integration of various knowledge sources.

(Potential) Users and Usage Scenarios

This includes a list of potential users or user groups and a description of each usage scenario. These scenarios should be described from the potential user who may report from own experiences: In what situation occurred a need for such a system (better search for information, information distribution etc.)? How did they proceed without it? How would they like to be supported? The usage scenarios sketch the point of view of each individual user, which may vary between extreme degrees. Those views give interesting input to the structure of the ontology. The descriptions of the hindering blocks include also important hints for the design of the ontology based system. The acquisition of the usage scenarios is done via structured or informal interviews. A common way of modeling usage scenarios in software engineering are use cases. In particular they help to identify stakeholders and to clarify their roles in the scenario.

Competency Questions

The usage scenarios (see above) describe the real existing domain of the targeted system. They deliver information about concepts and their relations which have to be modelled in the target ontology. To derive that information out of the use cases, the ontology engineer has to transform the scenarios in detailed competency questions (CQ) (Uschold & Grueninger, 1996). This represents an overview of possible queries to the system, indicating the scope and content of the application or domain ontology (*cf.* Section 2.3). Figure 3.5 shows an example of a competency questionnaire in Word[3].

Supported Applications

Draft of the ontology based knowledge management application and its system and software environment, which links the Knowledge Meta Process again to software engineering (*cf.* Section 3.1). The ontology engineer may here as well use the task analysis from the feasibility study as an input source to describe the proposed system and analyze the role of the ontology. The draft must also deliver a clear picture about the ontology interface to the user and answer the following question: what part of the ontology, namely concepts and relations are visible to the user and how does he use them? If the application runs several times on different hosts, one might want to keep track of the different locations to enable separate update processes in the maintenance phase.

3.4.2 Outcome

The outcome of this phase is (beside the ontology requirement specification document (ORSD)) a semi-formal description of the ontology, *i.e.* a graph of named nodes and (un-)named, (un-)directed edges, both of which may be linked with further descriptive text *e.g.* in form of mind maps like shown in the following section on tool support (*cf.* Figure 3.7).

[3]During later stages we developed a plugin for OntoEdit that allows for capturing of CQs within the ontology engineering environment. Thereby the references from extracted concepts and relationships can be stored along with an ontology to keep traceability. This might be helpful in later stages of the development, *e.g.* to show in which context a concept appeared. In Section 3.4.4 a screenshot of a completed questionnaire in OntoEdit is shown.

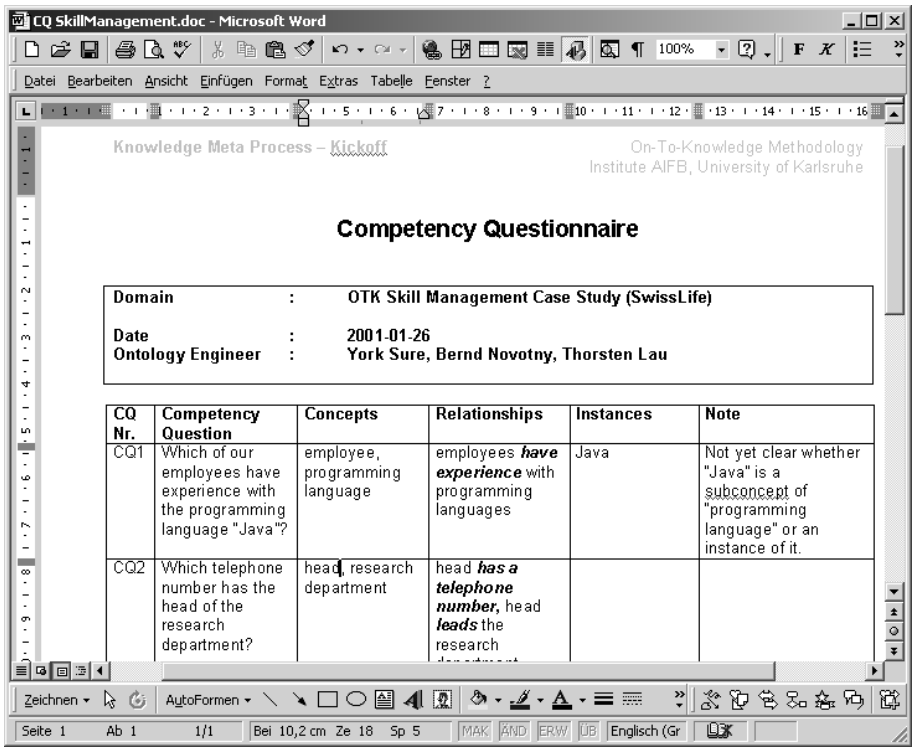

Figure 3.5: Competency questionnaire @ Swiss Life

3.4.3 Decision

If the requirements are sufficiently captured, one may proceed with the next phase refinement. This decision is typically taken from ontology engineers in collaboration with domain experts. "Sufficiently" in this context means, that from the current perspective there is no need to proceed with capturing or analyzing knowledge. However, it might be the case that in later stages one recognizes that the requirements are not sufficient anymore. Therefore the whole ontology development process is typically cyclic.

3.4.4 Tool Support

OntoEdit

To operationalize a methodology it is desirable to have tools that reflect and support all steps of the methodology and guide users step by step through the ontology engineering process. Along with the development of the methodology we therefore extended the core functionalities of OntoEdit by two plug-ins to support first stages of the ontology development, viz. OntoKick and Mind2Onto (*cf. e.g.* (Sure et al., 2002a))[4].

OntoKick targets at (i) creation of the requirement specification document and (ii) extraction

[4]Describing the plug-in framework is beyond the scope of this document, it is described in (Handschuh, 2001). In a nutshell, one might easily expand OntoEdit's functionalities through plug-ins.

of relevant structures for the building of the semi-formal ontology description. Mind2Onto targets at integration of brainstorming processes to build relevant structures of the semi-formal ontology description. As computer science researchers we were familiar with software development and preferred to start with a requirement specification of the ontology, i.e. OntoKick. People who are not so familiar with software design principles often prefer to start with "doing something". Brain storming is a good method to quickly and intuitively start a project, therefore one also might begin the ontology development process with Mind2Onto.

OntoKick is an OntoEdit plug-in that extends the functionality of OntoEdit by support for requirements specification (describing the plug-in structure itself is beyond the scope of this paper). OntoKick allows for describing important aspects of the ontology, viz.: the domain and the goal of the ontology, the design guidelines, the available knowledge sources (*e.g.* domain experts, reusable ontologies etc.), the potential users, the use cases, and the applications supported by the ontology. OntoKick stores these descriptions along with ontology definitions.

As proposed by (Uschold & King, 1995) and mentioned above, we use competency questions (CQ) to define requirements for an ontology. Each CQ defines a query that the ontology should be able to answer and therefore defines explicit requirements for the ontology. Typically, CQs are derived from interviews with domain experts and help to structure knowledge. We take further advantage of using them to create an initial version of the semi-formal description of the ontology. Based on the assumption that each CQ contains valuable information about the domain of the ontology we extract relevant concepts and relations (see example below). Furthermore, OntoKick establishes and maintains links between CQs and concepts derived from them. This allows for better traceability of the origins of concept definitions in later stages (this is also planned in a future version for relationships and instances).

We illustrate the usage of CQs by an example from our case study. Figure 3.6 shows a screenshot of our ontology environment OntoEdit presenting a draft competency questionnaire from Swiss Life.

Mind2Onto is a plug-in for supporting brainstorming and discussion about ontology structures. Especially during early stages of projects in general, brainstorming methods are commonly used to quickly capture pieces of relevant knowledge. A widely used method are mind maps™ (Buzan, 1974), they were originally developed to support more efficient learning and evolved to a management technique used by numerous companies. In general, a mind map™ provides information about a topic that is structured in a tree. Each branch of the tree is typically named and associatively refined by it's subbranches. Icons and pictures as well as different colors and fonts might be used for illustration based on the assumption that our memory performance is improved by visual aspects. There already exist numerous tools for the electronically creation of mind maps™. Many people from academia and industry are familiar with mind maps™ and related tools – including potential ontology engineers and domain experts. Therefore the integration of electronic mind maps™ into the ontology development process is very attractive (*cf. e.g.* (Lau & Sure, 2002)).

We relied on a widely used commercial tool[5] for the creation of mind maps™. It has advanced facilities for graphical presentations of hierarchical structures, *e.g.* easy to use copy&paste functionalities and different highlighting mechanisms. It's strength but also it's weakness lies in the

[5] MindManager™ 2002 Business Edition, *cf.* http://www.mindjet.com

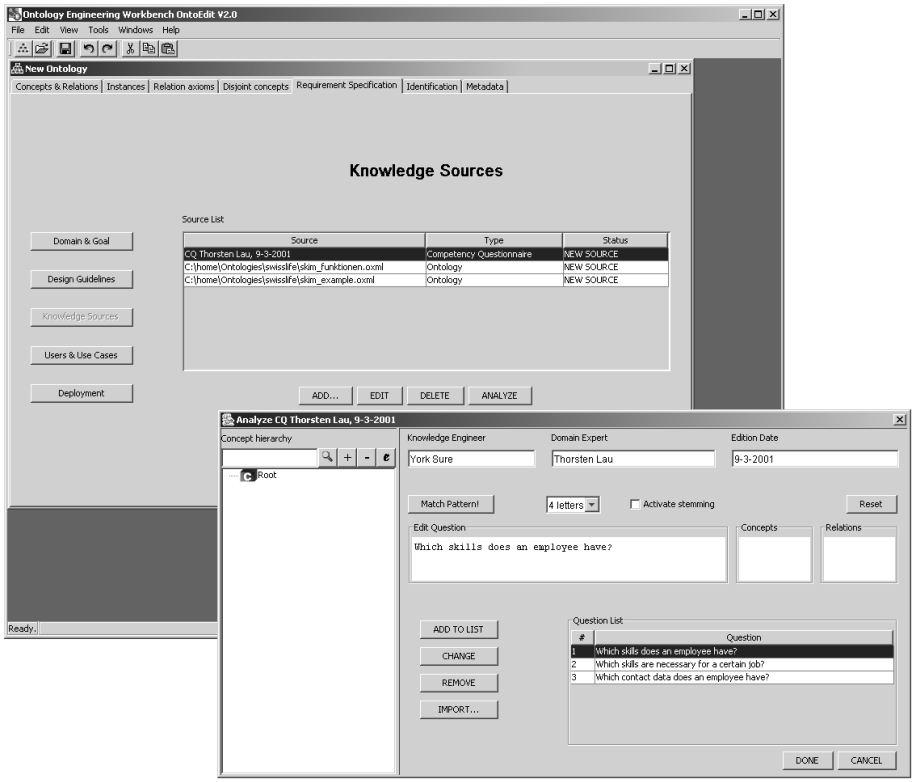

Figure 3.6: A Competency Questionnaire in the Ontology Engineering Environment OntoEdit

intuitive user interface and the simple but effective usability, which allows for quick creation of mind maps™ but lacks of expressiveness for advanced ontology modeling. By nature, mind maps™ have (almost) no assumptions for it's semantics, *i.e.* branches are somehow "associatively related" to each other. This assumption fits perfectly well during early stages of ontology development for quick and effective capturing of relevant knowledge pieces and makes the mind map™ tool a valuable add-on. Figure 3.7 shows a draft mind map™ for the BT case study.

Mind2Onto integrates the mind map™ tool into the ontology engineering methodology. Currently OntoEdit and the mind map™ tool interoperate through import and export facilities based on XML.

3.4.5 Experiences

Naming Concepts

Typically we tried to follow in the case studies the recommendation from ISO-704 (ISO 704, 1987) which is divided into three major sections: concepts, definitions and terms. Concepts are seen as units of thought that conforms to our view. The idea of a definition is that it fixes the concept in the proper position of the system. Terms represent natural language representations of concepts. ISO-704 recommends that one concept should ideally be represented by one natural language term. However, the automatically extraction of terms, *e.g.* as explored in the EnerSearch

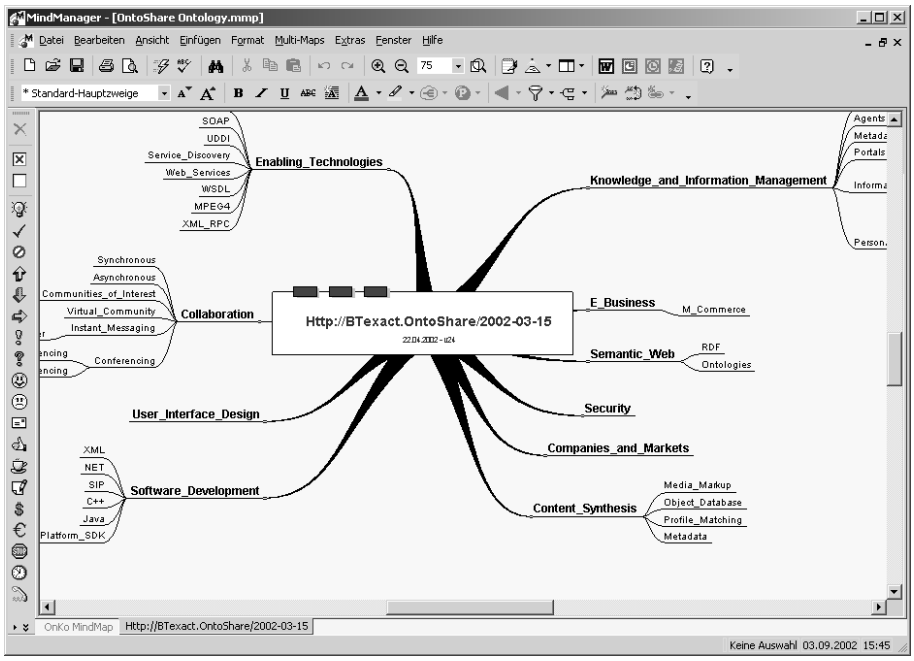

Figure 3.7: A mind map from the BT case study

case study, might lead to names that consist of more than one natural language term.

Manual *vs.* Automatic ontology Development

We discovered that a decision for a purely manual or (semi-) automatic ontology development approach is dependent on the domain type. Domains use vocabularies, which range from structured to unstructured. A structured vocabulary is common to all domain users and contains only a small set of synonyms and ambiguous terms, while unstructured vocabularies contain many words, which are used by the domain users in different ways and meanings.

For instance, the skills management domain is a more unstructured, a fuzzy one, because the vocabulary is shared by a lot of different people and departments with different backgrounds. For example, the private insurance department of Swiss Life has a completely different view and understanding on the terms “insurance premiums”, “customer” and “policyholder” as the group insurance department, which is based on different legislation. Therefore a common vocabulary does not exist and has to be agreed between the departments manually. This is the foremost reason why the automatic ontology development approach for the skills ontology development failed.

Information extraction tools typically are not able to match different words of a concept (synonyms) to a single term. It should be in mind that a structuring process takes much more time than the automatic approach based on the background to integrate different views of the departments into one common view, which tends to be complicated task depending on the people and the similarities between the departments.

Another lesson learned is that the domain has to be structured *in front of* the ontology modelling process. This is needed for an automatic ontology development as well as it is needed for

unifying different vocabularies of an unstructured, fuzzy domain. In the latter case the structuring process is manual work, which has to be done by domain experts like we have done *e.g.* for the skills management system.

A possible approach to speed up the development process is to build only a common top-level ontology for all departments and keep the individual parts of the departments. At first glance this is the desired solution. But, several disadvantages have to be considered. (i) A common understanding of the ontology can be lost. Each department still uses their vocabulary and the match between these is not documented. (ii) Some parts of the ontology can hold similar structures without noticing it, which is based on the separated and non coordinated development of the single parts. (iii) Top-level ontologies (recently known as foundational ontologies) are still a research topic. When it comes to implement them at an organizational level, and this is especially true for companies, time and money restrictions as well as cooperation difficulties between different departments might make the usage of such a top-level not possible. (iv) Handling multiple ontologies and mappings between them is still an immature area.

Abstraction Levels and Size

In our scenarios typically several people were involved in creating first draft versions of the ontology during early stages in the kick-off phase. Due to different backgrounds and mind models of the developers all of them had different abstraction levels. This made it difficult to merge them into a single coherent ontology.

Typical problems rose from differing numbers of branches per node and more general from how deep and wide the ontology is at all. Many iterations in the ontology development process were necessary to align *e.g.* the three skills ontologies in the Swiss Life case study (*cf.* Section 3.8) and finally to merge them together.

We stress that the ontology modelling process should start with a definition of the abstraction level, which is strongly dependent on the usage of the ontology. In the skills management scenario, we had to go down to a more precise level during the modelling process due to the high number of employees and the need to differentiate them for the intended project staffing functionality.

First feedback from test users of the skills management system and the underlying ontology showed that the skills trees are too large for browsing. Most users prefer shallow trees with a list of concepts on each node on a more abstract level. The reasons for this are the high time consuming usage of the deep ontology concepts and the amount of possible concepts, which can be chosen. The latter reason is based on the integrated skills ontology of all domains or departments.

A suggestion from the test users was to define user views on the ontology depending on the department where the user comes from. This would lead to much smaller skills trees and more usable ones in the according departments. Employees with broader skills can change from the department/domain view to the interdisciplinary view on the whole ontology. Though views are common to the database community, no concept for defining and using views on ontologies has been developed so far. As a summary, it is a challenging task to find the right balance between the requirements of different users of the skills management system. While some want to have a ontology as large as possible, the other ones feels comfortable with a small ontology.

3.5 Refinement

3.5.1 Knowledge Extraction

During the kick-off and refinement phase one might distinguish in general two concurrent approaches for modeling, in particular for knowledge extraction from relevant knowledge sources: top-down and bottom-up.

The usage scenario/competency question method follows usually a **top-down**-approach in modeling the domain. One starts by modeling concepts and relationships on a very generic level. Subsequently these items are refined. This approach is typically done manually and leads to a high-quality engineered ontology. Available top-level ontologies may here be reused and serve as a starting point to develop new ontologies. In practice we encountered a **middle-out** approach, *i.e.* to identify the most important concepts which will then be used to obtain the remainder of the hierarchy by generalization and specialization.

However, with the support of an automatic document analysis (*e.g.* with OntoExtract), a typical **bottom-up**-approach may be applied. There, relevant concepts are extracted semi-automatically from available documents. Based on the assumption that most concepts and conceptual structures of the domain as well the company terminology are described in documents, applying knowledge acquisition from text for ontology design seems to be promising.

Both approaches have advantages and drawbacks (*e.g.* the case studies at Swiss Life follow each a different path, *cf. e.g.* Section 3.8). The competency questions lead to a more detailed description of the problem area at hand. This supports the fine tuning of the ontology. On the other hand this gathering of several views is likely to be never complete and might not focus on the documents available. Semi-automatic text extraction is usually not able to produce high-level quality but delivers instead a more complete list of relevant concepts. So, the top-down-approach meets the representation of the ”information demand“ better than the bottom-up-approach with automatic analysis of documents, what itself supports a better representation of the ”information supply“.

A promising approach might be to combine both approaches. An automated extraction might be used as a starting point that is further refined by manual efforts of ontology engineers. However, the current drawback is that there is no tool support available for keeping the references between an extracted and the further refined version. This is a mandatory requirement for the maintenance of such a combined solution.

We propose that ontology engineers should include various knowledge sources depending on their availability and their reliability (see above) and use each time the more applicable method to extract relevant knowledge from the sources.

3.5.2 Formalization

To formalize the initial semi-formal description of the ontology we firstly form a taxonomy out of the semi-formal description of the ontology and add relations other than the “is-a” relation which forms the taxonomical structure. This knowledge about the domain is captured from domain experts in the previous mentioned competency questions or by using brainstorming techniques (*cf.*

Section 3.4.

The ontology engineer adds different types of relations as analyzed *e.g.* in the competency questions to the taxonomic hierarchy, *e.g.* in OntoEdit. However, this step is cyclic in itself, meaning that the ontology engineer now may start to interview domain experts again and use the already formalized ontology as a base for discussions. It might be helpful to visualize the taxonomic hierarchy and give the domain experts the task to add attributes to concepts and to draw relations between concepts (*e.g.* we presented them the taxonomy in form of a mind map™ as shown in the previous section). The ontology engineer should extensively document the additions and remarks to make ontological commitments made during the design explicit.

Depending on the application that has to be supported one has to choose an appropriate representation language. One should notice that formal representation languages typically differ in their expressive power and tool support for reasoning. The ontology engineer has to consider the advantages and limitations of the different languages to choose the appropriate one for the application.

Concerning OIL one has to consider three aspects:

- OIL is too less expressive. Many things cannot be expressed in it but could be easily expressed in a rule-based language like F-logic (*cf.* (Kifer et al., 1995)) oriented on reasoning over instances. The initial approaches to the semantic web were oriented around this paradigm (*cf.* (Heflin et al., 1999; Decker et al., 1999)).
- OIL is too expressive. OIL is logic based and OIL is description logic based. Many people find it difficult or not worth while to express themselves within logic. For example, non of the standard ontologies in electronic commerce or widely used ontologies such as Wordnet 6 make use of any axiomatic statements. Mostly they are simple taxonomies enriched by attributes in the best case. Description logic adds a specific feature: Concept hierarchies do not need to be defined explicitly by can be defined implicitly by complex definitions of classes and properties. Many people may find it easier to directly define *is-a* relationships instead of enforcing them by complex and well-thought axiomatic definition of classes and properties.
- Ontologies should not be based on formal logic. People with a background in databases wonder in general whether axioms, *i.e.*, complex logical statements should be part of an ontology. They tend to be application specific and very difficult to exchange and reuse in a different context. Spoken frankly, most of our experience conform with this statement.

3.5.3 Cyclic Approach

The refinement phase is closely linked to the evaluation phase. If the analysis of the ontology in the evaluation phase shows gaps or misconceptions, the ontology engineer takes these results as an input for the refinement phase. It might be necessary to perform several (possibly tiny) iterative steps to reach a sufficient level of granularity and quality.

3.5.4 Outcome

The outcome of this phase is the "target ontology", that needs to be evaluated in the next step.

3.5.5 Decision

The major decision that needs to be taken to finalize this step is whether the target ontology fulfills the requirements captured in the previous kickoff phase. Typically an ontology compares the initial requirements with the current status of the ontology. This decision will typically be based on the personal experience of ontology engineers. As a good rule of thumb we discovered that the first ontology should provide enough "flesh" to build a prototypical application. This application should be able to serve as a first prototype system for evaluation.

3.5.6 Tool Support

Sesame

Obviously the target ontology should be stored electronically. Sesame (*cf.* Section 2.5) not only provides a storage facility for RDF(S) ontologies, but also a query engine on top of the storage facility making Sesame also suitable for the later application of ontologies (*cf.* Section 3.7).

OntoExtract and OntoWrapper

OntoExtract and OntoWrapper from CognIT (*cf.* Section 2.5) provide support for semi-automatically extraction of relevant concepts and relations between them enabling a bottom-up approach as sketched in Section 3.5.1). The advantage of this approach is that the maintenance burden is reduced significantly in comparison to the manual development of ontologies.

OntoEdit

OntoEdit is the core tool to develop and maintain ontologies and offers as a main functionality to model ontologies on a conceptual level. To formalize this conceptualization into different representation languages like RDF(S) or OIL, we provide several export filters to the state-of-the-art ontology languages.

To minimize the gap between development, storage and application of ontologies, we connected OntoEdit via the **SesameClientPlugin** with Sesame (*cf.* Figure 2.11). Thereby one might directly up- and download ontologies from OntoEdit into Sesame.

Ontology engineering is by nature a cooperative task, typically involving ontology engineers and domain experts (*cf.* (Sure et al., 2002a)). The previously mentioned MindManager 2002 tool (*cf.* Section 3.4.4) is designed to support collaboration *e.g.* through small icons that have a specific meaning. By attaching these icons to certain branches, one might express certain agreements on the content like "This is ok." or "Here we need to elaborate more." etc.. We found this functionality very helpful (especially during the kickoff phases of the Swiss Life and BT case

studies) and decided to close the gap between the MindManager 2002 and OntoEdit beyond the previously mentioned Mind2Onto plugin functionalities.

The **OntoSkin** plugin for OntoEdit provides the possibility to mark concepts (and in a later development stage also relations and instances) with icons, *e.g.* the icons of the MindManager 2002. Figure 3.8 shows an example of how the MindManager 2002 icons are also used in OntoEdit. However, this plugin is currently in a very early stage of development.

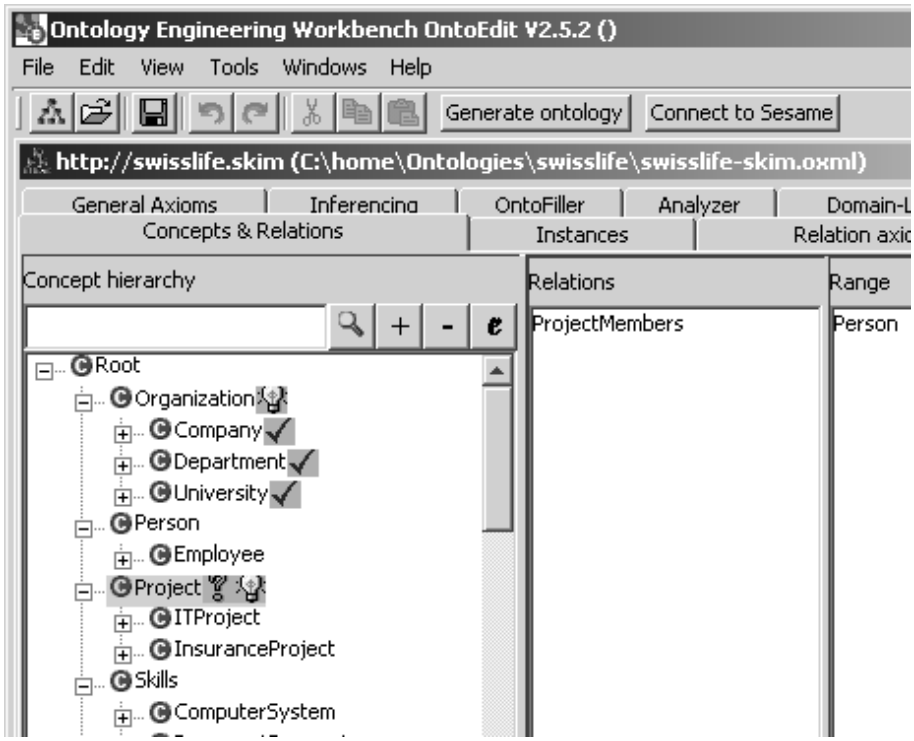

Figure 3.8: Using icons for collaboration with OntoSkin

The **OntoFiller** plugin (*cf.* Figure 3.9) addresses a specific requirement from the Swiss Life case study, the need for multi-lingual ontologies. It allows users to easily "fill in" external representations for concepts. The main purpose is to support users during the translation of an ontology into different languages (e.g. German, English, French and Italian - as needed in the SwissLife case study). The basic functionality consists of the generation of language-specific views on the ontology. Users can easily identify not yet translated concepts (or relations) and fill in external representations for various languages. Additionally, this plugin contains a translation support on top of the free German-English dictionary LEO (http://www.leo.org). A user can request "translation hints" from this online dictionary for concepts and relations. They consist of translations for a chosen concept or relation from German to English (and vice versa). A user then might choose from the received hints the appropriate translation for his purpose. As a side effect this plugin allows for viewing and editing the namespaces used for concepts and relations.

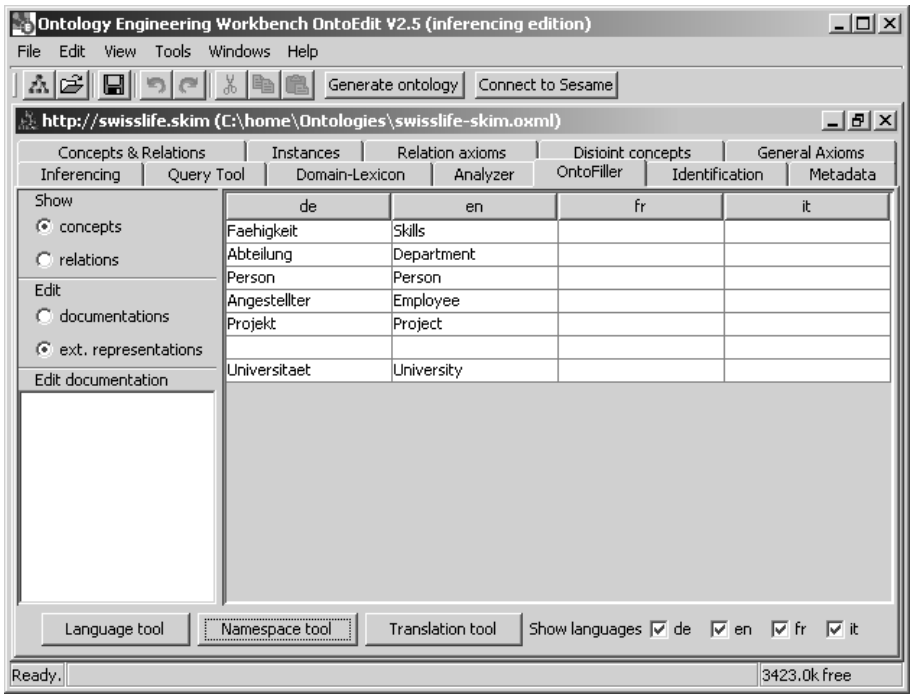

Figure 3.9: Inserting multi-lingual external representations for concepts and relations with OntoFiller

3.5.7 Experiences

Concepts *vs.* Instances

A typical problem area known from expert systems is the border between concepts and instances. Though experienced knowledge engineers with a practical background prefer not to distinguish between concepts and instances during the modelling process at all (ask, *e.g.* Hans Akkermans), our underlying architecture made it necessary to make this distinction upfront. As mentioned above it is difficult to distinguish between leaf nodes of the concept hierarchy and instances.

Our lesson learned in this area from the skills management case study at Swiss Life (see also Section 3.8) is that it is a good rule of thumb to count the possible instances and decide if this will be enough to justify a concept or not. In the case of the skills "Java" and "jdk1.3" we assumed that not so many employees will use "jdk1.3" so that our decision was to take "Java" as leaf node. In general it should be discussed close together with the definition of the abstraction and depth/width of the ontology, which builds a trade-off. Though there exists the simple heuristic to take every leaf node as an instance, this seems not to be correct for all use cases. But at all it is strongly dependent on the intended usage of the ontology, which determines the boarder.

De-contextualization of Concepts

Continuing the discussion of the ontology modelling for the skills management case study, a further difficulty emerged with the "de-contextualisation" of ontology concepts. We had the problem

that the ontology developers used concept names, such as “basics” in the ontology. For a human reader the meaning of such a concept is only comprehensible if its super-concept is known. When browsing the ontology this is no problem, because the super-concept is visible. But when a user selects this element as a skill then just “basics” occurs in his homepage, *i.e.* de-contextualiziation of this concept is made. The same problem occurred with “informatics” which occurs as a sub-concept of “skill”, “function” and “education” resulting in three different meanings of the concept (“informatics-skills”, “informatics-function” and “informatics-education”). It is no solution to force the ontology developers to use concept names that include the context of this concept. This would result in very long concept names. Furthermore, the engineers often forget about this problem and it is very hard to explain them why the concept name has to show the whole context. Therefore, we decided to “re-contextualize” the concept in the homepage by showing the path to the root concept of the ontology. So far we made good experiences in the skills management case study and the EnerSearch case study, where Spectacle also shows the navigational paths.

3.6 Evaluation

To describe the evaluation task, we cite (Gomez-Perez, 1996): “to make a technical judgement of the ontologies, their associated software environment, and documentation with respect to a frame of reference... The frame of reference may be requirements specifications, competency questions, and/or the real world.“

Because of the size of ontologies, their complexity, their formal underpinnings and the necessity to come towards a shared understanding within a group of people, ontologies are still far from being a commodity. Developing and deploying large scale ontology solutions typically involves several separate tasks and requires applying multiple tools, *e.g.* like the OTK tool suite. Therefore pragmatic issues such as *e.g.* interoperability and scalability (*cf.* (van Harmelen et al., 2001)) are key requirements if industry is to be encouraged to take up ontology technologies rapidly.

A systematic evaluation of ontologies and related technologies might lead to a consistent level of quality and thus acceptance by industry. For the future, this effort might also lead to standardized benchmarks and certifications. We therefore developed an evaluation framework for ontologies and related technologies[6].

We distinguish the following three types:

3.6.1 Technology-focussed Evaluation

Our evaluation framework for technology-focussed evaluation consists of two main aspects: (i) the evaluation of properties of ontologies generated by development tools, (ii) the evaluation of the technology properties, i.e. tools and applications which includes the evaluation of the evaluation tool properties themselves. In an overview these aspects are structured as follows in Table 3.1.

[6]This work is aligned with current efforts of the EU IST thematic network OntoWeb, *cf.* http://www.ontoweb.org, including numerous OTK partners, on evaluation of ontology related technologies. In particular an OntoWeb-SIG3 workshop is held in conjunction with the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002), *viz.* the workshop on “Evaluation of Ontology-based Tools (EON) 2002”, to be held in Siguenza (Spain) on 30th September 2002, *cf.* http://km.aifb.uni-karlsruhe.de/eon2002.

Ontology Properties	Technology Properties
Language conformity (Syntax)	Interoperability (e.g. Semantics)
Consistency (Semantics)	Turn around ability
	Performance
	Memory allocation
	Scalability
	Integration into frameworks
	Connectors and interfaces

Table 3.1: Evaluation Framework

For ontologies generated by the development tools the language conformity and consistency may be checked.

Language conformity means that the syntax of the representation of the ontology in a special language is conform to a standard. Such a standard is either a well-documented standard defined by a standardization body or it is an industrial standard mostly given by a reference implementation. So in the first case the outcome of an ontology tool must be checked with respect to the syntax definition and in the second case it must be tested using the reference implementation.

Evaluation of **consistency** means to what extent the tools ensure that the resulting ontologies are consistent with respect to their semantics, *e.g.* that different parts of the ontology representation do not contradict.

Ontology properties may be evaluated using the ontologies only, *i.e.* without having tools, *e.g.* for development, themselves available.

In contrast to that for the following second block of properties the tools are examined.

Interoperability means how easy it is to exchange ontologies between different tools. This includes such aspects as ”is a tool able to interpret the outcome of another tool in the same way?”. This is more than only checking the language conformity because it examines whether different tools interpret the same things in the same way. Often things can be represented in the same language in different ways.

Turn around ability means that the outcome of a tool is represented to the user in the same way again later on. *E.g.* a value restriction may be represented as a range restriction or by a constraint. If the tool shows that as a range restriction it should not show it as a constraint the next time it reads the same ontology.

Performance especially concerns the runtime effort of the tools, *e.g.* how much time is needed for solving a special inference task (not discussed here), for storing ontologies etc. Benchmark tests must be developed to evaluate these performance issues. For these benchmarks reference ontologies, reference ontology classes, reference tasks and reference task classes may be very helpful. This also refers to scalability (see below).

Memory allocation means how much memory is needed by the tools to handle ontologies. Similarly to the performance evaluation benchmarks must be available to test memory allocation.

For performance evaluation as well as for memory allocation it must be clarified what does the "size" of an ontology or the complexity of a task mean and which parameters influence this size in what way etc. This also refers to scalability (see below).

Scalability evaluates the performance and memory behavior of the tools with respect to increasing ontologies and tasks. It examines questions like "how increases a linear growth of ontologies the amount of memory allocated by the tool?".

Integration into frameworks means how easy it is to switch between such tools. For instance it is not very convenient that for a switch between tools it is necessary to store the ontology, to transform it afterwards with a different tool into another language which is a precondition to load it with the other tool. Entirely integrated environments similar to well-known programming environments must be the goal for ontology development tools.

Last but not least, the **connectivity** to other tools is important. This concerns (i) the connectors *to* other tools and (ii) the connectors *from* other tools to the tool.

3.6.2 User-focussed Evaluation

The framework shown above concentrates on the technical aspects of ontologies and related ontologies. However, the aspect of user-focussed evaluation remains open. We therefore present further steps for ontology evaluation that particularly make use of already available resources from earlier steps in the methodological framework.

Ontology engineers need to check, whether the target ontology itself suffices the ontology **requirements specification document** (*cf.* Section 3.4) and whether the ontology based application supports or "answers" the **competency questions**, analyzed in the kickoff phase of the project.

Therefore the ontology is tested in the target application environment. A **prototype** should already show core functionalities of the target system. **Feedback from beta users** of the prototype may be a valuable input for further refinement of the ontology.

A valuable input for refinement (and further maintenance) are **usage patterns** of the ontology. The system has to track the ways, users navigate or search for concepts and relations. With such an "ontology log file analysis" one may trace what areas of the ontology are often "used" and others which were not navigated. Less frequently used parts of the ontology should be monitored whether they are relevant for the application. High frequently used parts of the ontology might need to be expanded. However the ontology engineer should carefully evaluate the usage patterns before she updates the ontology.

The most important point from our perspective is to evaluate whether users are satisfied by the KM application. More specific, we evaluated **whether the ontology based technologies are at least as good as already existing technologies** in the EnerSearch case study and some results are described in Section 3.6.8.

3.6.3 Ontology-focussed Evaluation

Beside the above mentioned process oriented and pragmatic evaluation methods, there exist also formal evaluation methodologies for ontologies. One of the most prominent is the OntoClean

methodology (*cf. e.g.* (Guarino & Welty, 2002)), which is based on philosophical notions. It focuses on the cleaning of taxonomies and e.g. is currently being applied for cleaning the upper level of the WordNet taxonomy (*cf.* (Gangemi et al., 2002)). Core to the methodology are the four fundamental ontological notions of *rigidity*, *identity*, *unity* and *dependence*. By attaching them as meta-relations to concepts in a taxonomy they are used to represent the behavior of the concepts. I.e. these meta-relations impose constraints on the way subsumption is used to model a domain (cf. (Guarino & Welty, 2000)). We can only briefly and simplified sketch the methodology (please note that *property* is used here in our sense of "concept"):

Rigidity is defined based on the idea of essence. A *property* is essential to an individual if and only if necessarily holds for that individual. Thus, a *property* is rigid (+R) if and only if is necessarily essential to all its instances. A *property* is non-rigid (-R) if and only if it is not essential to some of its instances, and anti-rigid ($\sim$R) if and only if it is not essential to all its instances.

An **identity** criterion (IC) is carried by a *property* (+I) if and only if all its instances can be (re)identified by means of a suitable "sameness" relation. Additionally, a *property* supplies an identity criterion (+O) if and only if such criterion is not inherited by any subsuming *property*.

An individual *X* is constantly **dependent** on *Y* if and only if, at any time, *X* cannot be present unless *Y* is fully present, and *Y* is not part of *X*. A *property* is constantly dependent if and only if, for all its instances, there exists something they are constantly dependent on.

Unity is defined by saying that an individual is a whole if and only if it is made by a set of parts unified by a relation *R*. A *property P* is said to carry unity (+U) if there is a common unifying relation R such that all the instances of *P* are wholes under *R*. A *property* carries anti-unity ($\sim$U) if all its instances can possibly be non-wholes.

Based on these meta-relations OntoClean classifies concepts into categories (Sortal, Non-sortal, Role etc.). E.g., a concept that is tagged with "+O +I +R" is called a "Type". Beside these meta-relations OntoClean contains rules that can be applied to evaluate the correctness of a given taxonomy. For instance, a rule suggested in OntoClean is "a property carrying anti-unity has to be disjoint of a property carrying unity". As a consequence, "a property carrying unity cannot be a subclass of a property carrying anti-unity" and "a rigid property and an anti-rigid property are ever disjoint", to name but a few.

3.6.4 Cyclic Approach

The phases "Evaluation – Refinement – Evaluation" may need to be performed in iterative cycles to reach a sufficient level of quality.

3.6.5 Outcome

The outcome of this phase is an evaluated ontology, ready for the roll-out into a productive system. However, based on our own experiences we expect in most cases several iterations of "Evaluation – Refinement – Evaluation" until the outcome supports the decision to roll-out the application.

3.6.6 Decision

The major decision that needs to be taken for finalizing this phase is whether the evaluated ontology fulfills all evaluation criteria relevant for the envisaged application of the ontology.

3.6.7 Tool Support

OntoEdit

Tool support for the evaluation of ontologies seems to be a natural task for ontology engineering environments like OntoEdit. However, when we began to develop plugins we realized that we need advanced inferencing support for performing these tasks. Due to the close cooperation with Ontoprise GmbH (Germany), we decided to develop reference plugins on top of their inference engine Ontobroker. The implementations described in the following are only partially developed within the On-To-Knowledge project, but fit as an additional piece in the overall picture.

A motivational whitepaper for the workshop on "Evaluation of Ontology-based Technologies (EON2002)"[7] illustrates two possible OntoEdit plugin implementations of the evaluation framework, viz. **OntoAnalyzer** and **OntoGenerator**. To guarantee a tight integration into the development process, both reference implementations are realized as plug-ins for OntoEdit. Each of the two plug-ins addresses different aspects of the evaluation criteria presented in framework. OntoAnalyzer focuses on evaluation of ontology properties, in particular language conformity and consistency. OntoGenerator focuses on evaluation of ontology based tools, in particular performance and scalability. We refer to (Angele & Sure, 2002) for further details.

Beside the basic functionality of OntoEdit to inspect and edit ontologies, we started to implement the **OntoClean** methodology as a plugin for OntoEdit. To implement the OntoClean methodology as a plugin in OntoEdit[8], we formalized the meta-relations and classifications as a "meta ontology" that can be used to classify concepts of an ontology. We modelled both, the "meta ontology" and an example ontology (the example is taken from (Guarino & Welty, 2002)) that has to be evaluated, in OntoEdit and specified each concept of the regular ontology, i.e. all subconcepts of "Entity", as an instance of the top-level concept "Property" of the meta ontology through an axiom in F-Logic (Kifer et al., 1995):

$$\texttt{FORALL}\ \ A\ \ A : Property \leftarrow A :: Entity.$$

Figure 3.10 shows the subsequent steps: (1) model the ontologies, (2) fill the meta relations with values (i.e. tag the concepts with "carryR" (+R) etc.) and (3) specify the definitions and constraints from OntoClean as axioms. One can now ask queries in F-Logic by using an attached inference engine (in our prototype plugin we use Ontobroker, *cf.* (Fensel et al., 2000a)) to find inconsistencies according to the OntoClean methodology.

[7] The workshop is held in conjunction with the EKAW'02, further information can be found at http://km.aifb.uni-karlsruhe.de/eon2002.

[8] There is also the group from the Artificial Intelligence Laboratory of the Technical University of Madrid (UPM) working on the integration of the philosophically oriented OntoClean (Guarino & Welty, 2002) methodology with the process oriented METHONTOLOGY (Gomez-Perez, 1996) by extending the WebODE (Arprez et al., 2001) ontology development environment (cf. http://www.ontoweb.org/workshop/ontoweb2/slides/ontocleansig3.pdf)

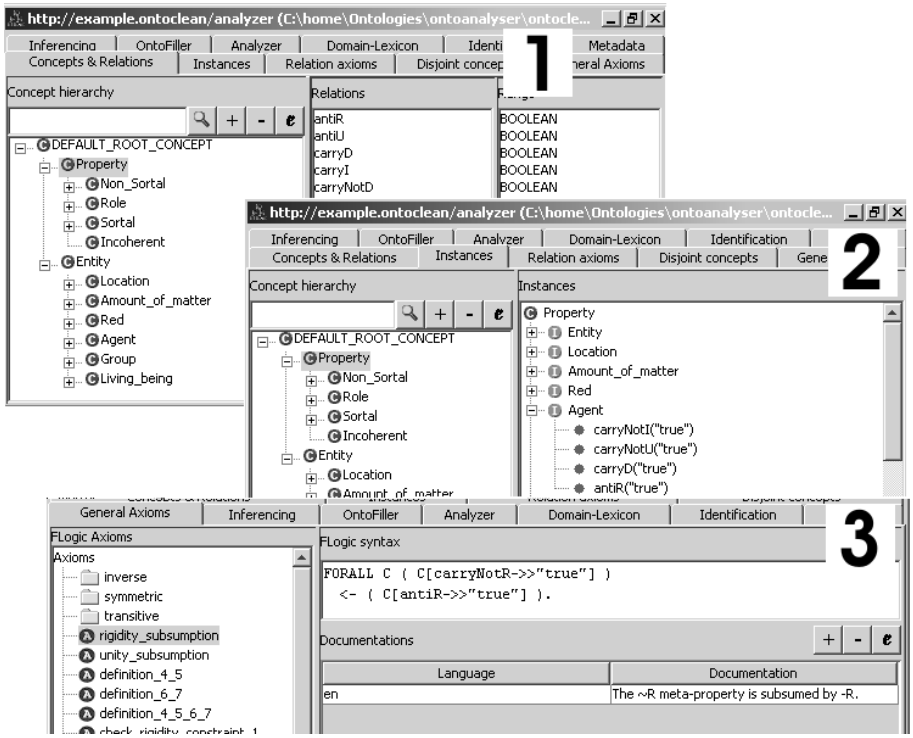

Figure 3.10: Implementation of OntoClean in OntoEdit

Further information on how OntoEdit supports ontology engineering by inferencing (including the here mentioned implementation of the OnotClean methodology) can be found at (Sure et al., 2002b). Ontobroker as a backend inference engine allows for additional advanced functionalities in OntoEdit, especially the evaluation of axioms. Having BOR as a dedicated DAML+OIL reasoner available now we plan to enable in OntoEdit similar functionalities as the one presented here for DAML+OIL (instead of F-Logic):

Analysis of Typical Queries. For this purpose, the Ontology engineer may interactively construct and save instances and axioms into modules. OntoEdit contains a simple instance editor that the ontology engineer can use to create test sets. The test set can be automatically processed and checked for consistency. As soon as the ontology proceeds into the evolution phase and needs changes to remain up-to-date, these test sets may be re-used for checking validity of the ontology.

Error Avoidance and Location. While the generation and validation of test cases allows for detection of errors, it does not really support the localization of errors. The set of all axioms, class and instance definitions express sometimes complex relationships and axioms often interact with other axioms when processed. Thus it is frequently very difficult to overview the correctness of a set of axioms and detect the faulty ones.

In principle there exist three types of problems with axioms:

- Axioms contain typing errors like variables not specified by a quantifier, typos in concept

names or relationship names etc.

- Axioms contain semantic errors, *i.e.* the rules do not express the intended meaning.
- Performance issues, like axioms defined such that evaluation needs a lot of time, which is not always easily recognizable by the user.

In order to avoid problems, OntoEdit offers several means:

1. Some axiom definitions may be generated by asserting through clicks that relations or concepts belong to particular types. OntoEdit allows for defining several properties of relationships by clicking on the GUI, *viz.* symmetry, transitivity and composition of relations.
2. For other types of axioms a graphical rule editor is available which avoids syntactical errors, delivers axioms which are optimal in their performance (as seen in isolation from other axioms) and supports users not familiar in F-Logic.
3. Third, there are axioms that cannot be specified by either 1. or 2. For them, OntoEdit provides at least syntax highlighting in order to support the user avoiding syntactical errors.

In order to locate problems, OntoEdit takes advantage of the inference engine Ontobroker itself, which allows for introspection and also comes with a debugger. Axioms are operationalized by posing queries (*e.g.* on the test cases specified as seen above). Based on queries one may pursue several alternatives:

First, a very simple but effective method to test axioms with test cases is to switch off and switch on axioms or parts of the axiom premises. The different answers from Ontobroker then allow to draw conclusions about possible errors.

Second, for a given query the results and their dependencies on existing test instances and intermediate results may be examined by visualizing the proof tree. This proof tree shows graphically which instances or intermediate results are combined by which rules to the final answers. Thus the drawn inferences may be traced back to the test instances and semantic errors in rules may be discovered.

Third, the inference engine may be "observed" during evaluation. A graphical presentation of the set of axioms as a graph structure indicates which axiom is evaluated at the moment and also shows which intermediate results have already been created up to now and thus "have flown" in the axiom graph to other axioms. This also gives the user a feeling how much time it is needed to evaluate special rules.

An example is given by the two Figures 3.11 and 3.12. The former illustrates how F-Logic axioms can be specified in OntoEdit in the "General Axioms" plugin (the more advanced graphical rule editor is near completion). All specified axioms are listed on the left side, on the right side one may see a selected F-Logic axiom as well as its documentation in different languages. The latter shows the GUI of the "Inferencing" plugin for OntoEdit, that integrates Ontobroker into OntoEdit. On the left side each previously specified axiom can be switched on or off. On the right side one may enter an F-Logic query (here: give me all instances of the concept "Konfiguration") which is subsequently answered by Ontobroker. The results are presented below the query, here one might see that for each result item the name of an instance includes the corresponding namespace.

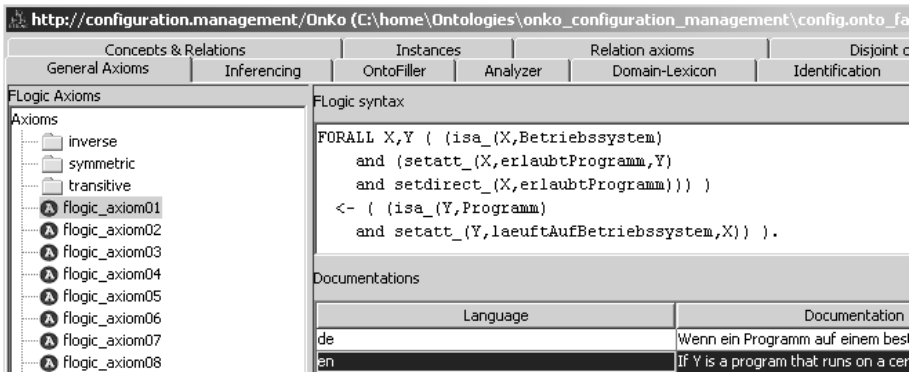

Figure 3.11: Specifying F-Logic axioms in OntoEdit

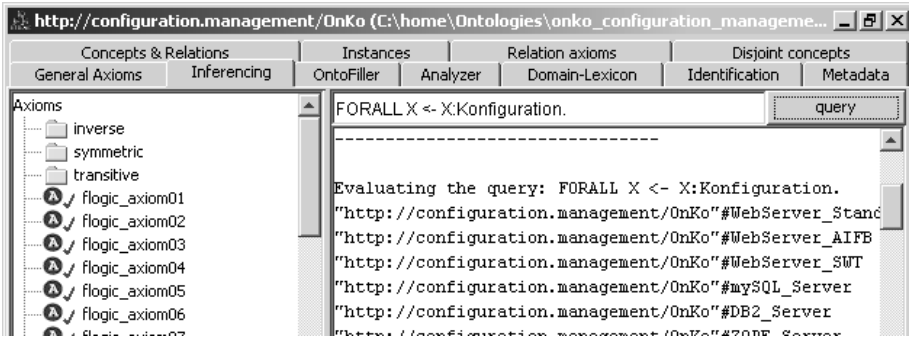

Figure 3.12: Inferencing with Ontobroker in OntoEdit

In the future, it is planned to take more care about the efficient construction of efficiently handable ontologies. For this purpose, OntoEdit will provide a profiler that will deliver statistics about evaluation times. And, as mentioned before, we plan to provide similar functionalities for DAML+OIL specific axioms.

3.6.8 Experiences

Lessons learned from the EnerSearch Experiment

To demonstrate the real value of Semantic Web methods we need to carry out field experiments. We have outlined in the EnerSearch case study a number of hypotheses that should be studied in representative case studies (*cf. e.g.* (Iosif & Sure, 2002)). We have also described what kind of variables have to be taken into account, how data collection, evaluation, experiment procedure, and system design can be done, and we have sketched the importance of the human side of information processing. At the time of this writing, we cannot yet give the final results of our field tests of Semantic Web tool use. However, several lessons and conclusions can be already be derived from our practical experiences so far:

- Semantic Web tool tests and case studies in the field require a very careful experiment design. Prospective test user groups and test tasks must be carefully balanced to allow

for adequate empirical-statistical testing of hypotheses that must be explicitly formulated in advance. Empirical data gathering in such experiments must be rich, including various qualitative methods – such as pre- and post-trial semi-open interviews, collecting verbal protocols during the experiment, onsite observation – as well as quantitative methods – for example electronic logging of actions and execution times, and statistical processing of resulting data.

- The case study validated the On-To-Knowledge approach of providing a set of tools and technologies for building customized knowledge management solutions, as opposed to providing a one-size-fits-all knowledge system. Tool integration, however, requires some additional components to the architecture that glue together the functionality of the components. There is also a need for a library for ontology transformations such as filtering and custom inferencing to fit the needs of the various subsystems.

- Building a search engine from an ontology using QuizRDF is a one-click process in contrast to creating an ontology-based presentation with Spectacle, which is a complex programming task. The QuizRDF search engine however cannot leverage ontological knowledge that goes beyond the common data model, while the Spectacle presentation can be custom tailored for the automatically extracted ontology (*e.g.* to present related concepts).

- Ontologies obtained through natural language processing are lightweight ontologies without a solid class hierarchy. This situation will be alleviated by supporting the automated ontology extraction through a repository of background knowledge that contains the domain information not found in the texts and guides the modelling process (*e.g.* by capturing design decisions such what is a class and what is an instance, *cf.*, *e.g.*, (Noy & McGuinness, 2001)). It is already possible, however, to capitalize on the advantages of automatically extracted ontologies that include, among others, a ranked selection of concepts, cross-taxonomical relationships (relations between concepts) and the automated mark-up of pages with concepts. This is particularly important for the Semantic Web where automated approaches and lightweight ontologies will prevail.

- The business case for ontology-based search and navigation is particularly strong for virtual enterprises, such as EnerSearch, whose main value driver is the creation and dissemination of (scientific) knowledge. For this kind of enterprise, the gains from employing ontologies can offset the significant technological risks involved with using advanced semantic technologies.

Virtual organizations provide a fertile test ground to validate the ideas that underlie knowledge management through Semantic Web methods. They have general characteristics that are such that semantic methods promise to be very beneficial. EnerSearch is such a knowledge-intensive virtual organization in which one of the main business ideas is to produce and transfer knowledge from researchers and experts to interested industries. So, internal, and even more importantly, external knowledge management is a key function where semantic methods can prove to be very helpful.

Evaluation of OTK Tools *vs.* Keyword-based Retrieval

The EnerSearch case study showed in detail the following (preliminary) results from analyzing the usage logfiles of the EnerSearch questionnaire system where we tracked performance and feedback of users. We refer to the final evaluation deliverable of EnerSearch to get the latest results (*cf.* (Iosif & Mika, 2002)).

The results here presented are based on a a small amount of data from the experiment, *i.e.* from seven test users, that partially answered our online questionnaire. Still, we are able to illustrate here some first impressions from the evaluation of our data. Each log entry contains the following items: question number, answer (text), name of the user, time duration for answering the question, the usergroup to which the user belongs, the tool used for getting the answer and how easy the user found to answer this question with the particular tool (on a scale from 1-"easy" to 5-"hard" plus 6-"I give up"). Until now we received 177 log entries from 7 test users. In our first analysis we concentrated on getting an impression for the two hypothesis:

1. Users will be able to complete information-finding tasks in less time using the ontology-based semantic access tools than with the current mainstream keyword-based free text search.

2. Users will make fewer mistakes during a search task using the ontology-based semantic access tools than with the current mainstream keyword-based free text search.

Figure 3.13 shows the calculated results for answering the question: "How relatively often did users give (W)rong, (R)ight or (N)o answers with each tool?". The figure shows the following preliminary results: For EnerSEARCHER, 23,19% of the questions answered in total (with EnerSEARCHer) were wrongly answered, 37,68% were answered right and in 39,19% of the cases the user gave up, thus resulting in having no answer at all for the question. For RDF-quiz, 10,20% of the questions were answered wrong, 57,14% were answered right and in 32,65% the user gave up. For Spectacle, 23,73% of the questions were answered wrong, 40,68% were answered right and in 35,59% cases the user gave up.

Thus, as a first result, our hypothesis 2 is supported by this result.

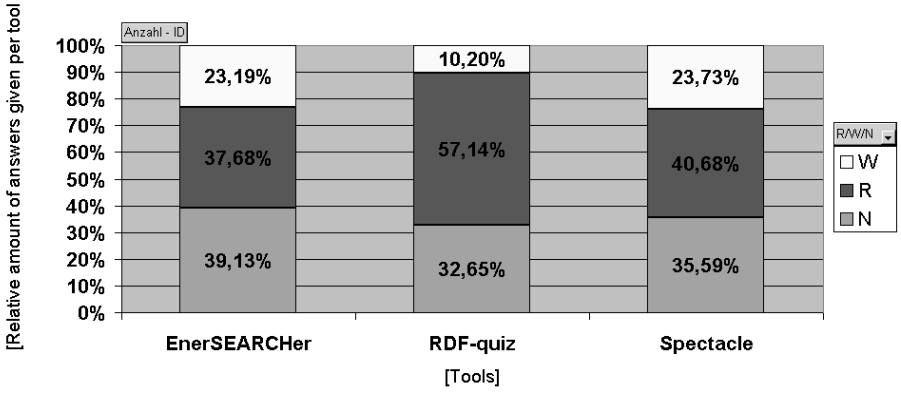

Figure 3.13: How relatively often did users give (W)rong, (R)ight or (N)o answers with each tool?

Figure 3.14 shows the calculated results for answering the question: "What relative average amount of time needed users for (W)rong, (R)ight or (N)o answering of one single question?". We highlight the most relevant detail of this figure (the reader might use the figure for further interpretations): To answer a question right, users needed in average the shortest amount of time with RDF-quiz (25,77%), followed by EnerSEARCHer (34,71%) and Spectacle (39,52%).

Thus, as a second result, our hypothesis 1 is partially supported by this result.

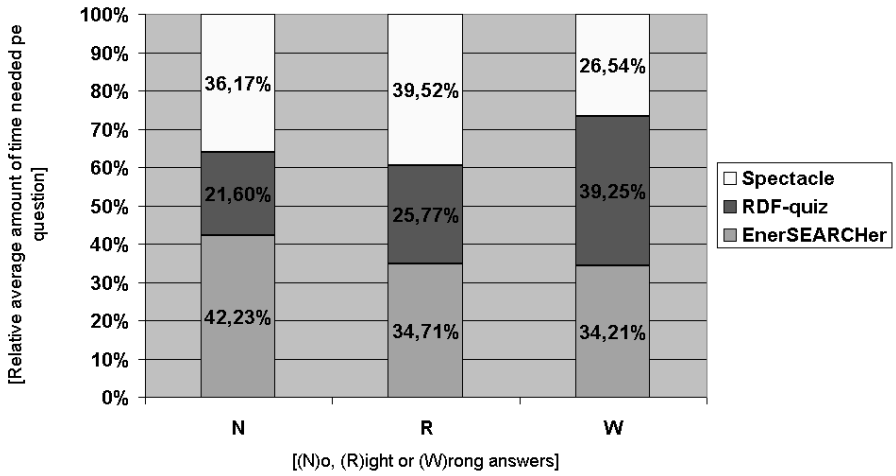

Figure 3.14: What relative average amount of time needed users for (W)rong, (R)ight or (N)o answering of one single question?

A careful look at the data revealed, that for some few entries in the logfile exist large durations for the answering of the questions, that are far beyond the "typical time" needed for finding an answer. These few entries have a significant influence on the average duration for answering questions and they exist for every given tool. We will investigate further to clarify whether users simply did other things than answering the questions during that time or what the reasons were for these long durations.

So far we only investigated in proving two hypothesis. However, the final results can be found in (Iosif & Mika, 2002). The ongoing tasks include the collection and complete evaluation of the data from the remaining test users (total of 45) as well as *e.g.* post-trials with the users to get feedback on the experiment as a whole. We will investigate which cognitive style users prefer and how that correlates with their usage of the tools, and, last but not least, how the clustering in the different user groups influence the results of our experiment.

Potential Threads for User focussed Evaluations

We identified in the EnerSearch case study several potential threads that might affect user focussed evaluations (*cf.* (Iosif & Sure, 2002; Sure & Iosif, 2002)). We list the threads and explain our strategy for avoiding them:

- **Users have no time:** One of the major problem with conducting a case study is to make the users interested in doing an evaluation. The test scenario will take 1 to 1.5 hours to finish.

Even though our test persons are well aware of the project there are always a great risk that the test users will have problem of finding the time to finish the tests. We designed our experiments from the user perspective. Instruction guide on how to use the tools were sent out in advance, in this way we minimized the chance of getting user problem with the actual test scenario.

- **Too few users for a comparative study:** An important issue is the choice of subjects who are going to participate. Practical concerns often constraints the experimentation possibilities, for example the accessibility and availability of certain types of subjects. We have identified several types of users and divided them into three different groups according their background and skills, particularly with respect to their familiarity and expertise with the EnerSearch web, knowledge management, knowledge acquisition tools and techniques. We ended up with 45 test persons that we believe is enough to do statistical comparative studies.

- **Too few comparable tools for a comparative study:** The EnerSearch consortium already have a tool, the EnerSEARCHer, that is an non ontology based search tool. It is a normal free text search tool. By combining two other ontology based tools, QuizRDF and Spectacle, we think that we have a good mix for the case study. With the ontology based search tool, QuizRDF, the user could start with simple queries consisting of only small number of search terms in order to get a picture on what kind of information is available in the database. The second ontology based search tool, Spectacle, present the information according to the inherent structuring that is offered by the ontology and that gives valuable context for the user.

- **Transfer error for a comparative study:** A problem within the subject experiments is that if one gives a subject the same exact search task to do with two different tools there will be most probably be a transfer error. This means that it is rather certain that they will be unlikely to repeat errors the second time they do the task, and that they will remember how they did something and will not need to figure it out the second time around. To avoid this transfer effect, we designed three different but comparable scenarios, each involving the same kind of knowledge acquisition task in the same domain but involving a different aspect of the knowledge base.

Formal Evaluation with OntoClean

The need for a formal evaluation of ontologies appeared very late in the On-To-Knowledge project due to the fact that during the setting up and implementation of the case study prototypes a formally evaluated ontology has not been the top priority, but rather to deal with organizational problems to get the case study up and running. The case studies do not have the top priority on creation of highly reusable and formally “clean” ontologies, but rather have as a top priority to get a KM application running that is accepted by its users.

Nevertheless for future extensions of the case study applications, a formally correct evaluation of the underlying ontologies seems crucial. When having a look at the methodology we were surprised that despite numerous projects that apply OntoClean there does not exist any well-known ontology editor that supports OntoClean. From our perspective an implementation in an ontology

engineering environment seems rather natural, therefore we started to implement the OntoClean methodology as a plugin for OntoEdit (Sure et al., 2002b).

However, we experienced that applying OntoClean requires persons that are trained in it – a skill that was certainly missing in the On-To-Knowledge project.

Combining ontologies with Data Mining

As described above, usage patterns derived from the navigational footprints users of an ontology based system leave are potentially valuable *e.g.* for evaluating how frequently certain ontology parts are used. In the Swiss Life case study on skills management it was planned to keep track of the usage patterns to provide suggestions which parts of an ontology might need modifications.

However, exploring the full potential of this issue is beyond the scope of the On-To-Knowledge project and is currently being tackled by another project, *viz.* the "SemiPort" project[9] that combines ontologies with data mining techniques to close the loops of engineering ontologies and using them (Gonzalez-Olalla & Stumme, 2002).

3.7 Application & Evolution

3.7.1 Application

The application of ontologies in productive systems, or, more specifically, the usage of ontology based systems, is being described in the following Chapter 4 that illustrates the "Knowledge Process".

3.7.2 Evolution

Organizational Aspects

We stretch that evolution of ontologies is primarily an organizational process. There have to be strict rules to the update/insert/delete processes of ontologies. We recommend, that the ontology engineer gathers changes to the ontology and initiates the switch-over to a new version of the ontology after thoroughly testing all possible effects to the application. Most important is to clarify *who* is responsible for maintenance and *how* it is performed and in *which time intervals* is the ontology maintained.

There exist two possible strategies for maintenance of ontologies: the **centralized** and the **distributed** strategy. In a centralized ontology, one single entity (*e.g.* a person) is responsible for maintaining the whole ontology or specific parts of it. Any modification (typically update-insert-delete) has at least to be approved by this entity. In the distributed strategy all modifications are made as they appear to be necessary and valuable by involved individuals.

Typically two aspects strongly influence the decision for one or the other strategy: **quality and time**. The responsible entity for maintenance is able to enforce and to guarantee to a certain

[9] http://km.aifb.uni-karlsruhe.de/semiport/

degree a certain quality of the ontology through thoroughly testing and checking possible effects of a modification. In most cases these modifications will take long time until they appear in the ontology. The distributed strategy is typically vice versa. Modifications appear immediately in the ontology – but also the level of quality may not be guaranteed and may change drastically over time.

Change Management for ontologies

Change management is especially important when ontologies will be used in a decentralized and uncontrolled environment like the Web, where changes occur without co-ordination. We refer to the subsection on OntoView in Section 2.5 for a more detailed view on change management for Ontologies.

3.7.3 Cyclic Approach

The phases "Application & Evolution – Refinement – Evaluation – Application & Evolution" and, as mentioned before, "Evaluation – Refinement – Evaluation" may need to be performed in iterative cycles.

3.7.4 Outcome

The outcome of an evaluation cycle is an evolved ontology, *i.e.* typically another version of it.

Additionally, an outcome of the application of an ontology within an ontology based application are *e.g.* usage patterns, as mentioned before, that can be used to derive hints for necessary evolutions of the ontology.

3.7.5 Decision

The major decision to be taken is when to initiate another evolution cycle for the ontology. This heavily depends on the local settlements of the organizational aspects (*cf.* Section 3.7).

3.7.6 Tool Support

Ontology Middleware Module

Though versioning for ontologies is a rather premature research area, On-To-Knowledge provides as part of the Ontology Middleware Module (OMM) a change management module for the versioning of ontologies (*cf.* Section 2.5).

3.8 Example: Skills Management at Swiss Life

To illustrate the instantiation, we give an example of the whole Knowledge Meta Process instantiation of the skills management case study at Swiss Life (*cf. e.g.* (Lau & Sure, 2002)). Further details and more examples from the case studies can be found in the OTK case study deliverables.

3.8.1 Feasibility Study

For identifying factors which can be central for the success or failure of the ontology development and usage we made a requirement analysis (Novotny & Lau, 2000) of the existing skills management environment and evaluated the needs for a new skills management system. As actors and stakeholders for the skills management we identified mainly the human resources department as well as the management level of all other departments. After finding the actors and stakeholders in the skills management area, we named the ontology experts for each department, which are preferably from the associated training group of each department.

3.8.2 Kickoff

The departments private insurance, human resources and IT as three different domains were the starting point for an initial prototype. Therefore the task was to develop a skills ontology for the departments containing three trees, *viz.* for each department one. These three trees should be combined under one root with having cross-links in between. This root node is the abstract concept “skills” (which means in German ”Kenntnisse/Fähigkeiten”) and is the starting point to navigate through the skills tree from the top.

Additionally for the gap analysis two ontologies are needed with the functions and education of the employees. The function ontology contains job descriptions available at Swiss Life with the therefore needed skills. In the education ontology all certificates, diploma, etc. were included. The aim of developing the education and functions ontologies is to know the educational background of an employee, which can be matched to the requirements of the function the employee has or wants to have. In the latter case the gap analysis is a means for the further qualification so that the employee is able to change his function.

During the kick-off phase two workshops with three domain experts[10] were held. The first one introduced the domain experts to the ideas of ontologies. Additional potential knowledge sources were identified by the domain experts, that were exhaustively used for the development of the ontologies, *e.g.* a book of the Swiss Association of Data Processing (“Schweizerischer Verband für Datenverarbeitung”) describing professions in the computing area in a systematic way similar to an ontology.

Obviously, this was an excellent basis to manually build the skills ontology for the IT domain. First experiments with extracting an ontology semi-automatically by using information extraction tools did not satisfy the needs for a clearly structured and easy understandable model of the skills. The domain experts and potential users felt very uncomfortable with the extracted structures and

[10]Thanks to Urs Gisler, Valentin Schoeb and Patrick Shann from Swiss Life for their efforts during the ontology modelling.

chose to build the ontology by themselves "manually".

To develop the first versions of the ontologies, we used a mind mapping tool ("MindManager"). It is typically used for brainstorming sessions and provides simple facilities for modelling hierarchies very quickly. The early modelling stages for ontologies contain elements from such brainstorming sessions (*e.g.* the gathering of the semi-formal ontology description).

During this stage a lot of "concept islands" were developed, which were isolated bunchs of related terms. These islands are subdomains of the corresponding domain and are self-contained parts like "operating systems" as sub domain in the IT domain. After developing these concept islands it was necessary to combine them into a single tree. This was a more difficult part as assembling the islands, because the islands were interlaced and for some islands it was possible to add them to more than one other island, which implies awkward skills trees. These skills trees would effect confusion on the users side, because the integration of the concepts looks not natural for the user.

For each department one skills tree was built in separate workshops. A problem that came up very early was the question where to draw the line between concepts and instances. *E.g.* is the programming language Java instantiated by "jdk1.3" or is "jdk1.3" so generic that it still belongs to the concept-hierarchy? Another problem was the size of the ontology. What is the best depth and width of each skills tree? Our solution was, that it is dependent of the domain and should be determined by the domain expert.

As result of the kick-off phase we obtained the semi-formal ontology descriptions for the three skills trees, which were ready to be formalized and integrated into a single skills ontology. The skills trees reached at this stage a maturity that the combination of them caused no major changes for the single skills trees.

3.8.3 Refinement

During the refinement phase we formalized and integrated the semi-formal ontology descriptions into a single coherent skills ontology. An important aspect during the formalization was (i) to give the skills proper names that uniquely identify each skill and (ii) to decide on the hierarchical structure of the skills. We discussed two different approaches for the hierarchical ordering: we discovered that categorization of skills is typically not based on an `is-a`-taxonomy, but on a much weaker HASSUBTOPIC relationship that has implications for the inheritance of attached relations and attributes.

However, for our first prototype this distinction made no difference due to missing cross-taxonomical relationships. But, according to (Guarino & Welty, 2002), subsumption provided by `is-a` taxonomies is often misused and a later formal evaluation of the skills ontology according to the proposed OntoClean methodology possibly would have resulted in a change of the ontology.

In a second refinement cycle we added one more relation type, an "associative relation" between concepts. They express relations outside the hierarchic skills tree, *e.g.* a relation between "HTML" and "JSP", which occur not in the same tree, but correspond with each other, because they are based on the same content. "HTML" is in the tree "mark-up languages", while the tree "scripting languages" contains "JSP". This is based on the basic characteristics and the history of both concepts, which changed over time. But in reality they have a close relationship, which can

be expressed with the associative relation.

The other task in this phase was to integrate the three skills ontologies into one skills ontology and eliminate inconsistencies in the domain ontology parts and between them. Because the domain ontologies were developed separately, the merger of them caused some overlaps, which had to be resolved. This happened for example in the computer science part of the skills trees, where the departments IT and private insurance have the same concepts like "Trofit" (which is a Swiss Life specific application). Both departments use this concept, but each from a different view, the IT from the development and the private insurance from the users view. Additionally the personal skills of any employee are graded according to a generic scale of four levels: basic knowledge, practical experience, competency, and top specialist. The employees will grade their own skills themselves. In other companies (*e.g.* Credit Suisse, ABB and IBM), such an approach proved to produce highly reliable information.

At the end of the refinement phase the "target skills ontology" consisted of about 700 concepts, which could be used by the employees to express their skill profile.

3.8.4 Evaluation

The evaluation of the prototype and the underlying ontology was unfortunately skipped due to internal restructuring at Swiss Life which led to a closing down of the whole case study.

3.8.5 Application & Evolution

Still, we considered the following aspects for the evolution of our skills management application: The competencies needed from employees are a moving target. Therefore the ontologies need to be constantly evaluated and maintained by experts from the human resource department. New skills might be suggested by the experts themselves, but mainly by employees. Suggestions include both, the new skill itself as well as the position in the skills tree where it should be placed. While employees are suggesting only new skills, the experts decide which skills should change in name and/or position in the skills tree and, additionally, decide which skill will be deleted. This was seen as necessary to keep the ontology consistent and to avoid that *e.g.* similar if not the same concept appear even in the same branch. For each ontology (and domain) there should exist a designated ontology manager who decides if and how the suggested skill is integrated.

Chapter 4

Knowledge Process

This chapter describes each of the steps of the Knowledge Process (the "dark grey circle" in Figure 2.1), i.e. "Knowledge Creation", "Knowledge Import", "Knowledge Capture", "Knowledge Retrieval & Access" and "Knowledge Use".

Compared to the previously described Knowledge Meta Process, the core concern here is not the initial setting up of an application including ontology development, but rather the ongoing usage of the (ontology based) knowledge management application. The focus is not so much on giving a detailed guidance on how to perform each step, but rather on depicting important aspects of running a KM application in real life. Each step has different priorities in each OTK case study (we will explain the ones with the highest priority). We give examples from the case studies and we show how each step can be supported by different OTK tools.

4.1 Steps of the Knowledge Process

Once a KM application is fully implemented in an organization, knowledge processes essentially circle around the following steps (*cf.* Figure 4.1).

- *Knowledge creation* and/or *import* of documents and meta data, i.e. contents need to be created or converted such that they fit the conventions of the company, *e.g.* to the knowledge management infrastructure of the organization;
- then knowledge items have to be *captured* in order to elucidate importance or interlinkage, e.g. the linkage to conventionalized vocabulary of the company by the creation of relational metadata;
- *retrieval of* and *access to knowledge* satisfies the "simple" requests for knowledge by the knowledge worker;
- typically, however, the knowledge worker will not only recall knowledge items, but she will process it for further *use* in her context.

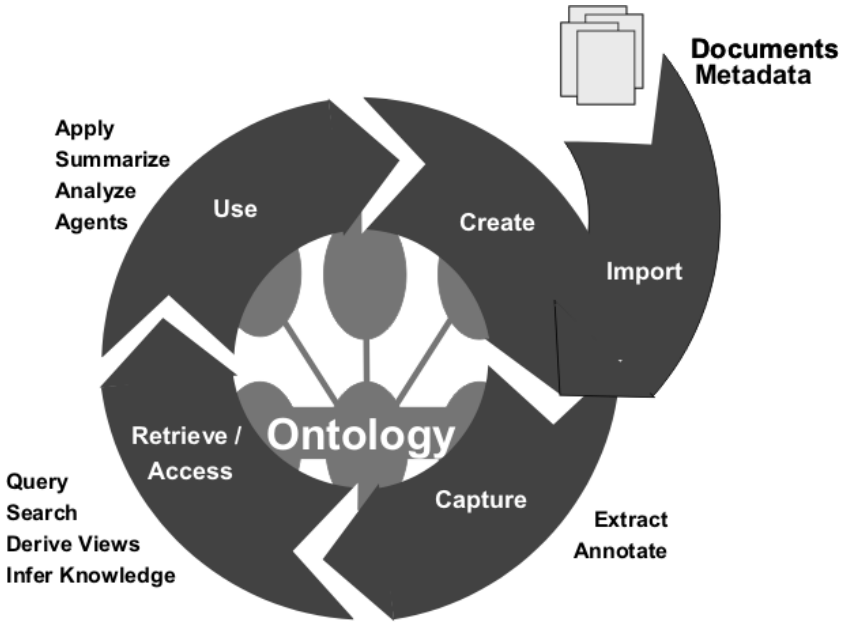

Figure 4.1: The Knowledge Process

4.2 Knowledge Creation

4.2.1 Formal vs. Informal Knowledge

Creation of computer-accessible knowledge proceeds between the two extremes of very formal and very informal knowledge. What is often overlooked is that comparatively deep coding can often be done without requiring any extra efforts. Business documents, in general, are not arbitrarily changing knowledge containers, but reasonably often come with some inherent structure, part of which is often required by quality management and, e.g., engineering requirements. Thus, a contribution to (Staab & O'Leary, 2000) proposed to embed the structure of knowledge items into *document templates*, which are then filled on the fly by doing daily work. The granularity of this knowledge then lies in the middle between the extremes of coarse representations of business documents only and an – for the purpose of KM – overly fine one, such as found in expert systems. Thus, one finds several degrees of formality between the two extremes of very formal and very informal knowledge. We compare some of them in Table 4.1.

In this comparison, we use the term "content-structured documents" to refer to e.g. XML structures that are tightly (sometimes explicitly, sometimes implicitly) linked to a domain model. For instance, XML-EDI documents come with a predefined structure alluding to a standard framework for exchanging data, such as invoices, healthcare claims, or project status reports. By document templates we refer to similar structures, which however come with a larger degree of freedom, including large chunks of informal knowledge items. One may note that these different degrees of formality are often combined, e.g. unstructured documents may have attached a form for adding Dublin Core meta data.

Careful analysis of the knowledge items in use allows for the possibility to add formal knowl-

Table 4.1: Degrees of formal and informal knowledge

Degree	Model	Interface	Example
Thoroughly Formal	Relational	Form Interface	Database interface
Formal	Content-structured document	Tight XML Structure	XML-EDI
Partially Formal	Document template	Loose XML Structure	Dublin Core templates
Informal	Free text	No predefined structure	ASCII text file

edge parts into the process of creating these documents, thus pushing the degree of formality slightly upwards without endangering the overall usage of the system, which could be incurred by an expert systems-like approach to Knowledge Management.

4.2.2 Example

Especially in the Swiss Life case study on skills management we integrated various degrees of formal knowledge: (i) we integrated existing databases (e.g. with personal contact data about employees) and (ii) applied ontology based templates to capture skill profiles in a structured way and (iii) provided a free-text facility for entering additional interests in an unstructured way (like e.g. hobbies).

4.3 Knowledge Import

4.3.1 ”Home-made” vs. Imported Knowledge

For many KM purposes the import of knowledge items into the KM system of the organization has the same or more importance than their creation within the organization. The overall situation is akin to data warehousing – only that the input structures are more varying and the target structures are much richer and more complex than this is the case for the standard data warehouse.

For imported knowledge accurate access to relevant items plays an even more important role than for “home-made” knowledge. The reason is that for home-made knowledge items, people may act as a backup index. This is not the case for recently imported knowledge that no one has seen yet. In fact, access studies to KM systems have shown that organizational memory parts that cover imported knowledge are less heavily exploited than those covering home-grown ones (OLeary, 1998) – though it seems implausible that they would contain less useful contents.

4.3.2 Example

For the virtual organization of EnerSearch knowledge import plays a major role. EnerSearch has a structure that is very different from a traditional research company. Research projects are carried out by a varied and changing group of researchers spread over different countries (Sweden, US, Netherlands, Germany, France). Many of them, although funded for their work, are not even

employees of EnerSearch. Thus, for its knowledge creation function EnerSearch is organized as a virtual research organization. Here it is important to notice that knowledge is not only created "within" EnerSearch but also imported from external researchers and consultants.

4.4 Knowledge Capture

4.4.1 Extraction and Annotation

Once that knowledge items have been created, but not yet, or only incompletely, captured apart from their context, e.g. from their database entries or their business document containers, the next process step is the capturing of their essential contents.

By this annotation process meta data are created that conform to the ontology and, hence, can be aligned with related information to yield analyzes and derivations. The origins of the meta data may be used to validate the overall information.

4.4.2 Example

We provide (i) common indexing and abstracting techniques with the OTK tool QuizRDF, and (ii) means to capture document excerpts as well as interlinkage between excerpts with the OTK tools OntoWrapper and OntoExtract.

4.5 Knowledge Retrieval and Access

4.5.1 Querying, Browsing and Navigation

Large parts of knowledge retrieval and access from an ontology-based organizational memory are performed through conventional GUI, exploiting means like information retrieval or taxonomy-enhanced database views. In addition, one may use the ontology to derive further views. In particular, we exploit the ontology for navigation purposes (*e.g.* with Spectacle). Thus, knowledge workers may explore what is in the organizational memory without being required to ask a particular question – which is often a hard task for newbies. Also, an ontology allows to derive additional links and descriptions by drawing inferences with appropriate inference engines like BOR, *e.g.* the ontology allows to derive state descriptions for points in time for which no explicit data exists, or it provides new hyperlinks that are not given explicitly. Thus, we may complete views without requiring that all information is given.

4.5.2 Example

The retrieval and access to knowledge is handled in general by the upper layer of the OTK tool suite, in particular by Spectacle and QuizRDF (*cf.* Section 2.5). As this is one of the core functionalities within On-To-Knowledge, it is naturally plays an important role in all case studies.

4.6 Knowledge Use

4.6.1 Add Value

Knowledge use deals with the most intricate points of knowledge management. It is the part that is most often neglected, because many KM systems assume that once some relevant document is found everything is done. Eventually, however, the way that knowledge from the organizational memory is used is quite involved. Therefore topics like proactive access, personalization, and, in particular, tight integration with subsequent applications play a crucial role for the effective re-use of knowledge. Very often it is not even the knowledge itself which is of most interest, but the derivations that can be made from the knowledge which is the added value on top of the existing knowledge.

In addition, usage data tells a lot about the organizational memory *and* about the organization. For instance, one may analyze which processes, customers and techniques are tied to core processes of the organization.

4.6.2 Example

The first item is *e.g.* tackled by OntoShare (used in the BT case study), that provides means for proactive access and personalization according to an underlying ontology.

The second item, usage data, is extensively explored in the EnerSearch experiment, *i.e.* the field study on evaluation. There, usage data is collected to evaluate compare the OTK tool suite (especially QuizRDF and Spectacle) *vs.* typical keyword based retrieval (EnerSEARCHer).

Chapter 5

Conclusion

5.1 The Problem

The Web and company intranets have boosted the potential for electronic knowledge acquisition and sharing. Given the sheer size of these information resources, there is a strategic need to move up in the data – information – knowledge chain.

5.2 Our Contribution

On-To-Knowledge takes a necessary step in this process by providing an innovative **tool suite** for semantic information processing for knowledge management and thus for much more selective, faster, and meaningful user access. This environment deals with three aspects:

- Acquiring ontologies and linking them to large amounts of data. For reasons of scalability, this process must be at least partially automated based on information extraction and natural language processing technology. To ensure quality, the process also requires human input in terms of building and manipulating ontologies based on ontology editors.
- Storing and maintaining ontologies and their instances. We developed an RDF Schema repository that provides database technology and simple forms of reasoning over Web information sources.
- Querying and browsing semantically enriched information sources. We developed semantically enriched search engines, browsing and knowledge sharing support that makes use of machine processable semantics of data.

The technology developed has been proven to be useful in a number of **case studies**, where it was applied according to the here presented methodology. We have shown how to improve information access in the large intranets of sizeable organizations. The technology has been used to facilitate electronic knowledge sharing and re-use for various knowledge management problems covering different cultures and different aspects:

- Introducing skills management and access to the "International Accounting Standards (IAS)" document as part of the company organizational memory at SwissLife.
- Enhancing the knowledge transfer to researchers and other company members in different disciplines and countries in the virtual organization of EnerSearch.
- Supporting the company internal community of knowledge sharing at BT.

This document, the OTK **methodology**, focusses on the identification, illustration and instantiation of the Knowledge Process and Knowledge Meta Process. To make it a self-contained document, we included the On-To-Knowledge building blocks, *i.e.* relevant aspects of knowledge management, ontologies and the overall project setting.

Beside others, one of the core contributions of this methodology is the linkage of the tool suite with the case studies by showing when and how to use available tools of the OTK tool suite during the process of developing and running the case studies. In a nutshell we have shown the following items:

- A process oriented methodology for introducing and maintaining ontology based knowledge management systems. Core to the methodology are Knowledge Processes and Knowledge Meta Processes. While Knowledge Meta Processes support the setting up of an ontology based application, Knowledge Processes support its usage.
- Tool support offered by the On-To-Knowledge tool suite for the process steps.
- Illustration of the process steps instantiations by numerous examples derived from the On-To-Knowledge case studies.

5.3 Outlook

We also encountered a number of shortcomings in our current approach. Ontologies help to establish consensual terminologies that make sense to computers and human beings. Computers are able to process information based on their machine-processable semantics. Human beings are able to make sense of that information based on their knowledge of real-world semantics. Building ontologies that are a pre-requisite for – and result of – the common understanding of large user groups is no trivial task.

A model or "protocol" for driving the network that maintains the process of evolving ontologies is the real challenge for making the semantic Web a reality. Most work on ontologies views them in terms of an isolated theory containing a potentially large number of concepts, relationships, and constraints that further detach formal semantics from them. To tap into the full potential advantages they offer for the Semantic Web, ontologies must be structured as interwoven networks that make it possible to deal with heterogeneous needs in the communication processes that they are supposed to mediate. Moreover, these ontologies need to shift over time because the processes they mediate are based on consensual representation of meaning.

It is the network of ontologies and their dynamic nature that make future research necessary. Actual challenges in the current work on ontologies are what glue ontology networks together

in space and time. Instead of a central, top-down process, we require a distributed process of emerging and aligned ontologies. Most existing technology focuses on building ontologies as graphs based on concepts and relationships. Our current understanding is far below par when it comes to proper methodological and tool support for building up networks, where the nodes represent small and specialized ontologies. This is especially true of the noisy and dynamically changing environment that the Web is and will continue to be.

These and other upcoming KM challenges are tackled in ongoing projects like “Ontologging”, “OntoWeb”, “WonderWeb” or “Semantic Web and Peer-to-Peer (SWAP)”.

Acknowledgements. The research presented in this paper greatly benefits from contributions of our colleagues at the Institute AIFB, University of Karlsruhe, and the closely related company Ontoprise GmbH. We thank especially Hans-Peter Schnurr (now Ontoprise GmbH) and Hans Akkermans (VU Amsterdam) for their seminal work while setting up the baseline version of this methodology. Last but not least we thank our colleagues of the On-To-Knowledge project for cooperative work, fruitful discussions, valuable contributions and, not to forget, the fun we had together.

Bibliography

Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O., & Sintek, M. (1998). Toward a technology for organizational memories. *IEEE Intelligent Systems*, pages 40–48.

Angele, J. & Sure, Y. (2002). EFFORT – evaluation framework for ontologies and related technologies. Technical report, Institute AIFB, University of Karlsruhe and Ontoprise GmbH.

Arprez, J., Corcho, O., Fernandez-Lopez, M., & Gomez-Perez, A. (2001). WebODE: a scalable workbench for ontological engineering. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP) Oct. 21-23, 2001, Victoria, B.C., Canada*.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 2001(5). available at http://www.sciam.com/2001/0501issue/0501berners-lee.html.

BOR (2002). http://www.ontotext.com/bor.

Brickley, D. & Guha, R. (1999). Resource description framework (RDF) schema specification. Technical report, W3C. W3C Proposed Recommendation. http://www.w3.org/TR/PR-rdf-schema/.

Broekstra, J., Fluit, C., & van Harmelen, F. (2000). The state of the art on representations and query languages for semistructured data. On-To-Knowledge deliverable D-8, AIdministrator Nederland b.v.

Broekstra, J. & Kampman, A. (2000). Query language definition. On-To-Knowledge deliverable D-9, AIdministrator Nederland b.v.

Broekstra, J. & Kampman, A. (2001). Sesame: A generic architecture for storing and querying RDF and RDF Schema. On-To-Knowledge deliverable D-10, AIdministrator Nederland b.v.

Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., & Horrocks, I. (2001). Enabling knowledge representation on the web by extending rdf schema. In *Proceedingso f the Tenth International World Wide Web Conference (WWW10)*, Hong Kong.

Buzan, T. (1974). *Use your head*. BBC Books.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., & Rice, J. (1998). OKBC: A programmatic foundation for knowledge base interoperability. In *AAAI/IAAI*, pages 600–607.

Corcho, O. & Gomez Perez, A. (2000). A roadmap to ontology specification languages. In Dieng, R. & Corby, O. (Eds.), *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00)*, volume 1937 of *LNAI*, pages 80–96. Springer Verlag.

DAML+OIL (2001). http://www.daml.org/2001/03/daml+oil-index.

Davies, J., Duke, A., & Stonkus, A. (2002). Ontoshare: Ontologies for knowledge sharing. In *RDF & Semantic Web Applications Workshop at the 11th International WWW Conference*, Hawaii, USA.

Decker, S., Erdmann, M., Fensel, D., & Studer, R. (1999). Ontobroker: Ontology based access to distributed and semi-structured information. In et al., R. M. (Ed.), *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic.

Dieng, R., Corby, O., Giboin, A., & Ribiere, M. (1999). Methods and tools for corporate knowledge management. *Int. Journal of Human-Computer Studies*, 51(3):567–598.

Duke, A. & Davies, J. (2001). Knowledge sharing facility. On-To-Knowledge deliverable D-12, BT.

Duke, A. & Davies, J. (2002a). Evaluation document. On-To-Knowledge deliverable D-26, BT.

Duke, A. & Davies, J. (2002b). Prototype. On-To-Knowledge deliverable D-25, BT.

Duke, A. & van der Meer, J. (2002). User profile construction. On-To-Knowledge deliverable D-14, BT and AIdministrator Nederland b.v.

Engels, R. & Bremdal, B. (2001a). CORPORUM: A workbench for the semantic web. In *Semantic Web Mining Workshop. PKDD/ECML - 01*, Freiburg, Germany.

Engels, R. & Bremdal, B. (2000). Information extraction: State-of-the-art report. On-To-Knowledge deliverable D-5, CognIT a.s.

Engels, R. & Bremdal, B. (2001b). Ontology extraction tool. On-To-Knowledge deliverable D-6, CognIT a.s.

Engels, R. & Bremdal, B. (2002). Ontowrapper. On-To-Knowledge deliverable D-7, CognIT a.s.

Fensel, D., Angele, J., Decker, S., Erdmann, M., Schnurr, H.-P., Studer, R., & Witt, A. (2000a). Lessons learned from applying AI to the web. *International Journal of Cooperative Information Systems*, 9(4):361–382.

Fensel, D., Hendler, J., Lieberman, H., & Wahlster, W. (Eds.) (2002). *Spinning the Semantic Web*. MIT Press, Boston.

Fensel, D., Horrocks, I., Harmelen, F. V., Decker, S., Erdmann, M., & Klein, M. (2000b). OIL in a nutshell. In Dieng, R. & Corby, O. (Eds.), *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00)*, volume 1937 of *LNAI*.

Fensel, D., Horrocks, I., van Harmelen, F., Broekstra, J., Crubezy, M., Decker, S., Ding, Y., Erdmann, M., Goble, C., Klein, M., Omelayenko, B., Staab, S., Stuckenschmidt, H., & Studer, R. (2000c). Ontology Language Version 1. On-To-Knowledge deliverable D-1, Vrije Universiteit Amsterdam.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–44.

Fensel, D., van Harmelen, F., & Horrocks, I. (1999). OIL: A standard proposal for the Semantic Web. On-To-Knowledge deliverable D-0, Vrije Universiteit Amsterdam.

Fensel, D. & van Harmelen, F. (2000). Project presentation On-To-Knowledge: Content-driven knowledge management tools through evolving ontologies. On-To-Knowledge deliverable D-33, Vrije Universiteit Amsterdam.

Fensel, D. (2001). *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, Berlin.

Fensel, D. (2002). Welcome to OIL. On-To-Knowledge deliverable D-2, Vrije Universiteit Amsterdam.

Fluit, C., ter Horst, H., & van der Meer, J. (2002). Visualization facility. On-To-Knowledge deliverable D-13, AIdministrator Nederland b.v.

Gangemi, A., Guarino, N., Oltramari, A., & Borgo, S. (2002). Cleaning-up WordNet's top-level. In *Proceedings of the 1st International WordNet Conference*, Mysore, India.

Gomez-Perez, A. (1996). A framework to verify knowledge sharing technology. *Expert Systems with Application*, 11(4):519–529.

Gonzalez-Olalla, J. & Stumme, G. (2002). Semantic methods and tools for information portals - the semiport project. In *Proceedings of the 2nd Workshop on Semantic Web Mining at ECML/PKDD-2002*, Helsinki, Finland.

Gruber, T. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, (43):907–928.

Guarino, N. & Welty, C. (2000). A formal ontology of properties. In Dieng, R. & Corby, O. (Eds.), *Knowledge Engineering and Knowledge Management: Methods, Models and Tools. 12th International Conference, EKAW2000*, pages 97–112. Springer Verlag.

Guarino, N. & Welty, C. (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65.

Guarino, N. (1998). Formal ontology and information systems. In *In Proceedings of FOIS'98 (Formal Ontology in Information Systems), Trento, Italy, 6-8 June 1998*. IOS Press.

Handschuh, S. (2001). Ontoplugins – a flexible component framework. Technical report, University of Karlsruhe.

Heflin, J., Hendler, J., & Luke, S. (1999). SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078, Institute for Advanced Computer Studies, University of Maryland.

Horrocks, I. & Hendler, J. A. (Eds.) (2002). *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*. Springer.

Horrocks, I. (1998). Using an expressive description logic: FaCT or fiction? In *Proceedings of KR 1998*, pages 636–649. Morgan Kaufmann.

Iosif, V. & Mika, P. (2002). EnerSearch virtual organisation case study: Evaluation document. On-To-Knowledge deliverable D-29, EnerSearch AB, Malmö, Sweden.

Iosif, V. & Sure, Y. (2002). Exploring potential benefits of the semantic web for virtual organizations. Submitted.

Iosif, V., Ygge, F., & Akkermans, H. (2001). EnerSearch virtual organisation case study: Requirements analysis document. On-To-Knowledge deliverable D-27, EnerSearch AB, Malmö, Sweden.

Iosif, V. & Ygge, F. (2002). EnerSearch virtual organisation case study: VE prototype. On-To-Knowledge deliverable D-28, EnerSearch AB, Malmö, Sweden.

ISO 704 (1987). Principles and methods of terminology. Technical report, International Standard ISO.

Jasper, R. & Uschold, M. (1999). A framework for understanding and classifying ontology applications. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99)*, Banff, Canada.

Jeen Broekstra, Arjohn Kampman, F. v. H. (2002). Sesame: A generic architecture for storing and querying rdf and rdf schema. In *(Horrocks & Hendler, 2002)*, pages 54–68.

Karp, P. D., Chaudhri, V. K., & Thomere, J. (1999). XOL: An XML-based ontology exchange language, version 0.3, july 1999.

Karvounarakis, G., Christophides, V., Plexousakis, D., & Alexaki, S. (2001). Querying rdf descriptions for community web portals. In *Proceedings of The French National Conference on Databases 2001 (BDA'01)*, pages 133–144, Agadir, Maroc.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843.

Kiryakov, A., Ognyanov, D., & Popov, B. (2002a). Ontology middleware implementation. On-To-Knowledge deliverable D-39, OntoText Lab.

Kiryakov, A., Simov, K. I., & Ognyanov, D. (2002b). Ontology middleware: Analysis and design. On-To-Knowledge deliverable D-38, OntoText Lab.

Klein, M. & Fensel, D. (2002). Ontoview – web-based ontology versioning. Technical report, Vrije Universiteit Amsterdam. Submitted, draft at http://www.cs.vu.nl/˜mcaklein/papers/ontoview.pdf.

Krohn, U. & Davies, J. (2001). Case study on call centers: Requirements analysis document. On-To-Knowledge deliverable D-24, BT.

Krohn, U. (2001). RQLvis and RDFferret. On-To-Knowledge deliverable D-11, BT.

Lau, T. & Sure, Y. (2002). Introducing ontology-based skills management at a large insurance company. In *Proceedings of the Modellierung 2002*, Tutzing, Germany.

Lopez, M. F., Gomez-Perez, A., Sierra, J. P., & Sierra, A. P. (1999). Building a chemical ontology using Methontology and the Ontology Design Environment. *Intelligent Systems*, 14(1).

Maedche, A., Staab, S., Studer, R., Sure, Y., & Volz, R. (2002). SEAL – Tying up information integration and web site management by ontologies. *IEEE-CS Data Engineering Bulletin, Special Issue on Organizing and Discovering the Semantic Web*.

Maedche, A. & Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2).

Maedche, A. (2002). *Ontology Learning for the Semantic Web*. Kluwer.

McGuinness, D., Fikes, R., Rice, J., & Wilder, S. (2000). An environment for merging and testing large ontologies. In *Proceedings of KR 2000*, pages 483–493. Morgan Kaufmann.

Morgenstern, L. (1998). Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103:1–34.

Novotny, B., Lau, T., Reich, J., & Reimer, U. (2001). Organizational memory – evaluation of case study prototypes. On-To-Knowledge deliverable D-21, Swiss Life.

Novotny, B. & Lau, T. (2000). Case studies on organizational memory. On-To-Knowledge deliverable D-19, Swiss Life.

Novotny, B. & Lau, T. (2001). Organizational memory – description of case study prototypes. On-To-Knowledge deliverable D-20, Swiss Life.

Noy, N. & Hafner, C. (1997). The state of the art in ontology design – a survey and comparative review. *AI Magazine*, 36(3).

Noy, N. & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.

Ogden, C. & Richards, I. (1923). *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Routledge & Kegan Paul Ltd., London, 10 edition.

O'Leary, D. & Studer, R. (2001). Knowledge management: An interdisciplinary approach. *IEEE Intelligent Systems, Special Issue on Knowledge Management*, 16(1).

O'Leary, D. (1998). Using AI in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, 13(3):34–39.

OLeary, D. (1998). Using ai in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, 13(3):34–39.

Ontology Middleware Module Demo (2002). http://omm.ontotext.com.

Ontology Middleware Module (2002). http://www.ontotext.com/omm.

Probst, G., Romhardt, K., & Raub, S. (1999). *Managing Knowledge*. J. Wiley and Sons.

Schnurr, H.-P., Sure, Y., Studer, R., & Akkermans, H. (2000). On-To-Knowledge Methodology — baseline version. On-To-Knowledge deliverable D-15, Institute AIFB, University of Karlsruhe.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., & Wielinga, B. (1999). *Knowledge Engineering and Management — The CommonKADS Methodology*. The MIT Press, Cambridge, Massachusetts; London, England.

Sesame Source Forge Project (2002). http://sourceforge.net/projects/sesame.

Sesame (2002). http://sesame.aidministrator.nl/.

Simov, K. & Jordanov, S. (2002). BOR: a pragmatic DAML+OIL reasoner. On-To-Knowledge deliverable D-40, OntoText Lab.

Staab, S. & O'Leary, D. (Eds.) (2000). *Bringing Knowledge to Business Processes. Papers from 2000 AAAI Spring Symposium*, Technical Report SS-00-03, Menlo Park, CA. AAAI Press.

Staab, S., Schnurr, H.-P., Studer, R., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems, Special Issue on Knowledge Management*, 16(1).

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., & Wenke, D. (2002a). OntoEdit: Collaborative ontology development for the semantic web. In *(Horrocks & Hendler, 2002)*, pages 221–235.

Sure, Y. & Iosif, V. (2002). First results of a semantic web technologies evaluation. In *Proceedings of the Common Industry Program at the federated event co-locating the three international conferences: DOA'02: Distributed Objects and Applications; ODBASE'02: Ontologies, Databases and Applied Semantics; CoopIS'02: Cooperative Information Systems DOA/ODBASE/CoopIS'02*, University of California, Irvine, USA.

Sure, Y., Maedche, A., & Staab, S. (2000). Leveraging corporate skill knowledge - from ProPer to OntoProPer. In Mahling, D. & Reimer, U. (Eds.), *Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM 2000)*, Basel, Switzerland.

Sure, Y., Staab, S., & Angele, J. (2002b). OntoEdit: Guiding ontology development by methodology and inferencing. In *Proceedings of the International Conference on Ontologies, Databases and Applications of SEmantics ODBASE 2002*, University of California, Irvine, USA.

Sure, Y. & Studer, R. (2001a). OntoEdit. On-To-Knowledge deliverable D-3, Institute AIFB, University of Karlsruhe.

Sure, Y. & Studer, R. (2001b). On-To-Knowledge Methodology — evaluated and employed version. On-To-Knowledge deliverable D-16, Institute AIFB, University of Karlsruhe.

Sure, Y. & Studer, R. (2002). On-To-Knowledge Methodology — expanded version. On-To-Knowledge deliverable D-17, Institute AIFB, University of Karlsruhe.

Sure, Y. (2002). On-To-Knowledge technical fact sheet for the OTK tool suite demo. On-To-Knowledge technical fact sheet, Institute AIFB, University of Karlsruhe.

Uschold, M. & Grueninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Sharing and Review*, 11(2).

Uschold, M. & King, M. (1995). Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, Montreal, Canada.

van Harmelen, F., Kampman, A., & Broekstra, J. (2001). Interoperability and scalability of On-To-Knowledge tools. On-To-Knowledge deliverable D-X1, Vrije Universiteit Amsterdam.

van Heijst, G. (1995). *The Role of Ontologies in Knowledge Engineering*. PhD thesis, Universiteit van Amsterdam.