Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften (Dr. rer. pol.)
von der Fakultät für Wirtschaftswissenschaften
der Universität Fridericiana zu Karlsruhe
genehmigte Dissertation.

Methodology, Tools & Case Studies for Ontology based Knowledge Management

von

Dipl.-Wi.-Ing. York Sure

Tag der mündlichen Prüfung: 19. Mai 2003
Referent: Prof. Dr. Rudi Studer
1. Korreferent: Prof. Dr. Karl-Heinz Waldmann
2. Korreferent: Prof. Dr. Robert Meersman

To my family.

Acknowledgements

This thesis is the result of my work as a research assistant at the Institute AIFB at the University of Karlsruhe, Germany. Many people supported me and made the successful completion of this thesis possible.

First of all, I would like to express my gratitude to my advisor, Prof. Dr. Rudi Studer. His decent and goal-oriented management allowed freedom in my research and, at the same time, gave me strong support. His trust in my abilities greatly enhanced my performance in the projects we were working in.

I would like to thank Prof. Dr. Robert Meersmann (Free University of Brussels), who was not only willing to serve on my dissertation committee as co-referent, but who previously gave very valuable and stimulating feedback on my work as part of his role as a reviewer for the On-To-Knowledge project. Furthermore, I thank Prof. Dr. Karl-Heinz Waldmann (University of Karlsruhe), who was willing to serve on my dissertation committee as co-referent, and also Prof. Dr. Wolffried Stucky (University of Karlsruhe), and Prof. Dr. Thomas Lützkendorf (University of Karlsruhe) who served on the examination committee.

The On-To-Knowledge project provided a frame for my research. Thanks to Prof. Dr. Dieter Fensel (University of Innsbruck), Prof. Dr. Frank van Harmelen (Free University of Amsterdam), and Prof. Dr. Hans Akkermans (Free University of Amsterdam). It would exceed the scope of these acknowledgements to mention every single project partner. It was a great pleasure working with you and I have learned a lot from you.

The OntoWeb thematic network was a perfect training ground for my personal communication and, of course, networking skills. Thanks to all active OntoWeb members for their collaboration and support and particularly to our partners at the Free University Brussels (VUB). The group headed by Prof. Dr. Robert Meersman was our official project partner for the portal workpackage and together we handled the distributed development of an integrated portal.

Thanks to the "OntoTeam Karlsruhe" that consists of all my colleagues at the Institute AIFB, the FZI and at Ontoprise GmbH, for providing a very fruitful and stimulating working atmosphere – even after work. I would like to mention some of them who influenced and enhanced my work.

Hans-Peter Schnurr (Ontoprise) provided the first stone on which this thesis was built, *viz.* the baseline version of the methodology. He deserves special gratitude for teaching me an unreplaceable "coolness in tricky business situations" and, of course, how to make real good presentations. Priv.-Doz. Dr. Steffen Staab (AIFB) significantly improved my scientific skills, *e.g.* by teaching me how to successfully write papers, and contributed many ideas to my thesis. Priv.-Doz. Dr. Gerd Stumme (AIFB) accompanied and supported me in the teaching tasks which are also an important part of the life at the Institute AIFB.

With Raphael Volz (AIFB), Daniel Oberle (AIFB) and Jens Hartmann (AIFB) I shared the AIFB workload in OntoWeb. While I typically covered the political and marketing aspects, they actually implemented our ideas and maintained the OntoWeb.org portal. Andreas Hotho (AIFB) was always a great help with hardware problems and especially during the last days of this thesis he strongly supported me in keeping my personal writing motivation on the required level.

Prof. Dr. Jürgen Angele (Ontoprise) was always open to explain me tricky details of F-Logic and Ontobroker. Henrik Oppermann (Ontoprise) never hesitated to discuss with me about deep ontological foundations, *e.g.* by clarifying the meaning of concepts like "Heideggern". Alexander Mädche (FZI) and Dirk Wenke (Ontoprise) initiated the development of OntoEdit, which is now being continued by the constant efforts of Dirk. Together with Siegfried Handschuh (AIFB), who invented and implemented the OntoMat-Plugin-Framework, they provided the core technology for making the implementation part of this work possible.

Gisela Schillinger and Susanne Winter were always helpful to minimize my administrative work and, not to forget, provided uncountable cups of coffee.

Numerous other people supported my work (among other things) with heavy implementation work. Particularly I would like to thank Sören Bergmann, Claus Boyens, Jeen Broekstra, Tomasz Gonsior, Arjohn Kampmann, Stefan Lenhart, Andreas Maier, Eddie Mönch, Borislav Popov, Heiko Rudat, Hans Trost and Markus Zondler.

My parents, Gudrun and Reinfried, and my sister, Katrin, receive my deepest gratitude and love for the many years of support and especially for always believing in me. Last, but not least, I thank my partner, Petra, and her son, Neil, for their love and understanding that strongly encouraged me and, in the end, made this thesis possible.

May 2003, Karlsruhe

York Sure

Contents

Acknowledgements v

I Foundations 1

1 Introduction & Overview 3
1.1 Introduction . . . 3
1.2 Contributions of This Thesis . . . 7
1.3 Reader's Guide . . . 9

2 Knowledge Management 11
2.1 Knowledge Transfer . . . 11
2.2 Building Blocks of Knowledge Management . . . 12
2.3 Knowledge Processes . . . 16
2.4 Knowledge Items & Meta Data . . . 16

3 Ontologies & The Semantic Web 21
3.1 History & Foundations . . . 21
3.2 Classification of Ontologies . . . 25
3.3 Communication with Ontologies . . . 27

II On-To-Knowledge Methodology 29

4 Knowledge Meta Process 31
4.1 Developing and Deploying Knowledge Management Applications . . . 31
4.2 Steps of the Knowledge Meta Process . . . 32
4.3 Feasibility Study . . . 34
4.4 Kickoff . . . 37
4.5 Refinement . . . 42
4.6 Evaluation . . . 46
4.7 Application & Evolution . . . 50

5 Knowledge Process **53**
5.1 Steps of the Knowledge Process 53
5.2 Knowledge Creation & Import 54
5.3 Knowledge Capture 57
5.4 Knowledge Retrieval & Access 59
5.5 Knowledge Use 61

III Tool Support **63**

6 Requirements for OEEs **65**
6.1 Motivation of Requirements (R1 – R5) 65
6.2 Ontologies in OntoEdit (R1) 68
6.3 Extensibility through the OntoMat Framework (R2) 71
6.4 Methodology Support (R3) 74
6.5 Dimensions of Collaboration (R4) 75
6.6 Theoretical & Practical Inferencing Issues (R5) 77

7 Kickoff Support **83**
7.1 Competency Questions (and other Requirements) 83
7.2 Brainstorming 86
7.3 External Support 90

8 Refinement Support **91**
8.1 Documentation & Translation 91
8.2 Distributed Engineering 93
8.3 External Support 98

9 Evaluation Support **101**
9.1 Scalability Evaluation 101
9.2 Guidelines Evaluation 104
9.3 Formal Evaluation 106
9.4 External Support 112

10 Application & Evolution Support **115**
10.1 Storage & Versioning 115
10.2 External Support 116

IV Case Studies **119**

11 On-To-Knowledge **121**

11.1 About the Project 122
11.2 Technical Architecture 123
11.2.1 OTK Tool Suite 124
11.2.2 OIL: Ontology Inference Layer for the Semantic Web 135
11.3 Configuration of Tools in the Case Studies 141
11.3.1 Tools @ Swiss Life 142
11.3.2 Tools @ BT 143
11.3.3 Tools @ EnerSearch 143
11.4 Skills Management @ Swiss Life 144
11.4.1 Introduction 144
11.4.2 Feasibility Study 145
11.4.3 Kickoff 149
11.4.4 Refinement 152
11.4.5 Evaluation, Application & Evolution 156
11.4.6 Main Lessons Learned @ Swiss Life 156
11.5 Community of Knowledge Sharing @ BT 158
11.5.1 Introduction 158
11.5.2 Feasibility Study 158
11.5.3 Kickoff & Refinement 160
11.5.4 User-Focused Evaluation 162
11.5.5 Application & Evolution 164
11.5.6 Main Lessons Learned @ BT 165
11.6 Virtual Organization @ EnerSearch 168
11.6.1 Introduction 168
11.6.2 User-Focussed Evaluation 169
11.6.3 Main Lessons Learned @ EnerSearch 178
11.7 Overall Lessons Learned in On-To-Knowledge 179

12 OntoWeb **181**
12.1 About the Thematic Network 181
12.2 SEmantic portAL (SEAL) of OntoWeb 183
12.2.1 Motivation for Semantic Portals 183
12.2.2 Web Information Integration 185
12.2.3 Web Site Management 186
12.2.4 Implementation of the OntoWeb Semantic Portal 187
12.2.5 Process Model 190
12.2.6 Future Directions 194
12.3 Evaluation Efforts 196
12.3.1 Workshop Description 196
12.3.2 Travelling Domain Description 197
12.3.3 Engineering the Model with OntoEdit 199

12.3.4 Main Lessons Learned in EON2002 209
12.3.5 Future Directions for EON Experiments 211

V Related Work & Conclusions 217

13 Related Work 219
13.1 Related Work on Methodologies 219
13.2 Related Work on Tools 221
13.3 Related Work on Case Studies 223

14 Conclusions 225
14.1 Summary 225
14.2 Impacts & Outlook 227

VI Appendix 231

A OntoEdit Outside 233

B OntoEdit Inside 249

C DAML+OIL features supported by OntoEdit 253

D Example Ontology 257

Bibliography 285

Index 311

List of Figures

2.1 Two orthogonal processes with feedback loops 16

3.1 Different kinds of ontologies and their relationships 26
3.2 The Meaning Triangle 27
3.3 Communication between human and/or software agents 28

4.1 Relevant processes for developing and deploying KM applications 32
4.2 The Knowledge Meta Process 33
4.3 CommonKADS worksheets for feasibility study 35

5.1 The Knowledge Process 54
5.2 Annotating documents with the OntoMat–*Annotizer* 58
5.3 Knowledge Retrieval with DOGMA in the OntoWeb Portal 60

6.1 Ontology development steps of the On-To-Knowledge Methodology . . . 66
6.2 Basic functionalities of OntoEdit 69
6.3 Modeling instances with OntoEdit 70
6.4 Plugin architecture of OntoEdit 74
6.5 Overview of the "methodology plugins" 75
6.6 Overview of "external plugins" 76
6.7 Overview of the "collaboration plugins" 77
6.8 Overview of the "inferencing plugins" 78

7.1 Capturing competency questions with OntoKick 85
7.2 Traceability of concepts with OntoKick 86
7.3 Example mind map about this work 88
7.4 Support discussions with Mind2Onto 89

8.1 Inserting multi-lingual external representations for concepts and relations with OntoFiller 92
8.2 Client-Server architecture of OntoEdit 94
8.3 Locked trees in OntoEdit 97

9.1 Generating ontologies with OntoGenerator 103
9.2 Combinations of OntoClean meta-relations 109
9.3 Ideal taxonomy structure 110
9.4 Implementation of OntoClean in OntoEdit 111
9.5 Deriving inconsistencies with the OntoClean Plugin 112
9.6 Visualizing ontologies in the Visualizer Plugin 113

10.1 Storing and versioning ontologies with the Sesame Plugin 116

11.1 OTK technical architecture 124
11.2 The query interface of *Quiz*RDF 126
11.3 Provision of navigational structures with Spectacle 127
11.4 Ontology development with OntoEdit 129
11.5 Features of the Knowledge Control System 130
11.6 The result of a comparison of two ontologies with OntoView 132
11.7 Sesame: repository for ontologies and data 134
11.8 OntoExtract: Automatic ontology generation 136
11.9 OntoWrapper: Information extraction 137
11.10The layered language model of OIL 140
11.11OIL illustration 140
11.12Covering of tools @ Swiss Life case study 142
11.13Covering of tools @ BT case study 143
11.14Covering of tools @ EnerSearch case study 144
11.15Roadmap for Skills Management @ Swiss Life 146
11.16Skills management prototype 148
11.17Competency questionnaire @ Swiss Life 150
11.18The skills management competency questionnaire in OntoKick 151
11.19Mind map of skills @ Swiss Life 153
11.20Communities of knowledge sharing with OntoShare @ BT 160
11.21Mind map @ BT 161
11.22EnerSEARCHer (free text) 174
11.23EnerSEARCHer results 175
11.24*Quiz*RDF (ontology based) 176
11.25Spectacle (ontology based) 176
11.26How relatively often did users give (W)rong, (R)ight or (N)o answers with each tool? 177
11.27What relative average amount of time needed users for (W)rong, (R)ight or (N)o answering of one single question? 178

12.1 SEAL conceptual architecture 185
12.2 The OntoWeb Semantic Portal 188

12.3 Publishing workflow in the OntoWeb portal (i) 192
12.4 Publishing workflow in the OntoWeb portal (ii) 193
12.5 MindMap of the travelling domain 200
12.6 Concept hierarchy 201
12.7 Relationships of Flight 202
12.8 Relationships for Plane 202
12.9 Relationships of Accomodation 203
12.10 "Abstract" *vs.* "concrete" concepts 203
12.11 Relationships of City 204
12.12 Instances of City and Attraction 205
12.13 Instances of Means of transport and Three Stars Hotel 205
12.14 Instances of Flight and Trip 206
12.15 Transitive relationships 206
12.16 Inverse relationships 207
12.17 Disjoint concepts 207
12.18 General axioms 208
12.19 Inferencing with Ontobroker in OntoEdit 209
12.20 Switching axioms off in OntoEdit 209

A.1 The main GUI of OntoEdit 234
A.2 File Menu 235
A.3 Insert a URI 235
A.4 Create a new ontology 236
A.5 Search for a specific concept 236
A.6 Concept context menu 237
A.7 Insert a new concept 237
A.8 Edit selected concept / concept properties 238
A.9 Relation context menu 239
A.10 Insert new relation 239
A.11 Context menu for instances 241
A.12 Editing an instance 242
A.13 Relation axioms tab 242
A.14 Disjoint concepts tab 243
A.15 Identification tab 244
A.16 Metatada tab 244
A.17 View menu 245
A.18 Options for plugins 246
A.19 Tools menu (i) 246
A.20 Plugin management during runtime 247
A.21 Tools menu (ii) 247

List of Tables

2.1 Dimensions of knowledge transfer 12
2.2 Approaching the Knowledge Process — two extreme positions 17

4.1 Technology-focussed evaluation framework 47

5.1 Degrees of formal and informal knowledge 55

6.1 Summary of the requirements for Ontology Engineering Environments . 67

7.1 OntoKick characterization 84
7.2 Mind2Onto characterization 87

8.1 OntoFiller characterization 92
8.2 Client-Server Framework characterization 93

9.1 OntoGenerator characterization 102
9.2 OntoAnalyzer characterization 105
9.3 OntoClean Plugin characterization 107

10.1 Sesame Plugin characterization 115

Part I

Foundations

"All men by nature desire knowledge."
— Aristotle

1 Introduction & Overview

In **this part** we provide the **foundations** of the presented thesis. We start with an introduction and an overview about the contributions. The following Chapter 2 introduces "knowledge" as a central concept and illustrates the combination of knowledge management and ontologies. This part ends with Chapter 3 where the concept "ontology" is approached in the scope of this work. Current trends of using such ontologies for KM motivate the need for the core contributions of this work, *viz.* an advanced methodology, tools and case studies for ontology based KM.

This chapter starts with an introduction and motivation in Section 1.1. Next, the contributions of this work are summarized in Section 1.2. The chapter ends with a reader's guide that sketches the overall structure of the work in Section 1.3.

1.1 Introduction

This thesis combines the topics **knowledge management** and **ontologies**. The driving force for combing the two topics is the growing importance of knowledge for societies and their economies. Both topics are now briefly introduced to motivate the contributions of this thesis described in the next section.

Our society changed from being an industrial society to being a knowledge society. This shift of paradigms enforced enterprises to act no longer based on purely tayloristic principles but rather as *"intelligent enterprises"* (*cf.* (Quinn, 1992)). **Knowledge** became the key economic resource, as Drucker pointed out:

"The basic economic resource – the means of production – is no longer capital, nor natural resources, nor labor. It is and will be knowledge."
(Drucker, 1993)

This so-called *"post-industrial revolution"* (*cf.* (Jacques, 1996)) focussed the view on knowledge as *"intellectual capital"* (*cf.* (Stewart, 1997)) that is a mission critical resource. Therefore, companies should have the same interest in managing their knowledge as in managing capital investment and working relationships (*cf.* (Edvinson & Malone, 1997)).

Knowledge management (KM) was born as significant corporate strategy to meet the new challenges. The history and the current status of KM is sketched by Kay:

> *"Knowledge management as an approach to business management has had a tumultuous history. It was born as a hip buzzword, was shunned as a second cousin to business process reengineering, and was for a time hijacked by software vendors. Despite this circuitous path, knowledge management is now well on the way to becoming a necessary component of every bottom-line-oriented company's long-term business strategy."*
> (Kay, 2003)

The main goal of typical current KM initiatives is to enable a better knowledge sharing. Drivers for the introduction of knowledge management were *e.g.* the potential for reduction of (i) costs for duplication of efforts, (ii) loss of knowledge when key people leave a company and (iii) time needed to find correct answers. This has led to many efforts for capturing, storing and making knowledge accessible. But, as Davenport and Prusak mention, sharing knowledge requires a common language:

> *"People can´t share knowledge if they don't speak a common language."*
> (Davenport & Prusak, 1998)

Successful KM strategies consist of building blocks for organization, people, technology and corporate culture (*cf.* (Albrecht, 1993; Schneider, 1996a), we will elaborate more on that in Section 2.2). In such a context, knowledge sharing is not only a matter of communication between people, but also between people and technology and between technology, *i.e.*, *e.g.*, software agents that communicate with people or each other. More general, agents (human and software agents) need to share their knowledge and require a common language. Thus, we generalize the quotation from above to *"Agents can't share knowledge if they don't speak a common language"*.

Ontologies were exploited in Computer Science to enhance knowledge sharing and re-use (*cf.*, *e.g.*, (Gruber, 1993; Fensel, 2001)). Firstly, they provide a shared and common understanding of knowledge in a domain of interest. Secondly, they capture and formalize knowledge by connecting human understanding of symbols with their machine processability. As such, ontologies act as a common language between agents.

The use of ontologies for knowledge management offers therefore great advantages. Numerous applications already exist (*cf.*, *e.g.*, (Sure & Schnurr, 2003)).

Common knowledge management applications make use of available technology that was originally developed for the World Wide Web, *e.g.* the now very popular corporate intranets. Similar to the Web they provide access to a large amount of information contained in documents, databases *etc.* and suffer from the same weaknesses, *e.g.*, among others,

(i) **searching information** often leads to irrelevant information,

(ii) **extracting information** is left to the burden of humans since software agents are not yet equipped with common sense and domain knowledge to extract such information from textual representations and they fail to integrate information distributed over different sources,

(iii) **maintaining** weakly structured text sources is a time-consuming and difficult task when such sources become large (*cf.* introduction of (Davies et al., 2002b)).

To overcome such weaknesses of the current Web, Berners-Lee and others envisioned the **Semantic Web**:

> *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation."*
> (Berners-Lee et al., 2001)

The Semantic Web extends the Web by adding machine-processable meta-information, aka meta data, to documents. The meta data explicitly define what the document is about. Thereby, ontologies provide the schema for meta data to make them re-usable and define their meaning.

The vision of the Semantic Web is now beginning to become reality. The vision (i) evolved into a research area, (ii) attracted numerous researchers and industrial members, and (iii) reached already a large visibility, *e.g.* by the **"First International Semantic Web Conference (ISWC 2002[1])"**, held 2002 in Sardinia, Italy (*cf.* (Horrocks & Hendler, 2002)).

The ideas and technologies of the Semantic Web have a large influence on current trends in knowledge management, *e.g.* visible in the subtitle (and, of course, the contributions) of the **"13th International Conference on Knowledge Engineering**

[1]ISWC 2002, see `http://iswc2002.semanticweb.org/`

and Knowledge Management (EKAW 2002[2]) – Ontologies and the Semantic Web", held 2002 in Sigüenza, Spain (*cf.* (Gómez-Pérez & Benjamins, 2002)).

[2]EKAW 2002, see `http://babage.dia.fi.upm.es/ekaw02/`

1.2 Contributions of This Thesis

The work presented in this thesis is mainly the result of research performed within the EU project On-To-Knowledge[3] and its "spin-off" thematic network OntoWeb[4]. In a nutshell, we will present some Semantic Web methods and technologies and show how they can be applied to practical knowledge management in corporate intranets and in the Web.

We tackle the following research questions:

1. **How do ontology based applications evolve?**
   a) What are the relevant processes for engineering and using these applications?
   b) What methodologies do already exist and how can they be integrated, adopted and extended?
2. **How can the engineering of ontology based applications be supported by tools?**
   a) What are the requirements for such tools?
   b) What kind of tool support is available?
3. **How can methodology and accompanying tools be applied to real problems?**
   a) What kind of problems are tackled?
   b) How are they solved?
   c) What are the lessons learned?

The contributions of this thesis cover a broad spectrum of aspects related to practical ontology based knowledge management. With respect to the questions raised above we provide the following answers:

1. **Methodology:** We present the novel On-To-Knowledge Methodology for introducing and maintaining ontology based knowledge management applications. Unlike most other (similar) methodologies, the On-To-Knowledge Methodology is independent of specific domain knowledge to make it as re-usable as possible. We distinguish two processes that should be kept separate in order to achieve a clear

[3]EU IST-1999-10132 project On-To-Knowledge, see `http://www.ontoknowledge.org/`
[4]EU IST-2000-29243 thematic network OntoWeb, see `http://www.ontoweb.org/`

identification of issues: whereas the first process addresses aspects of introducing a new KM application 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”). Our main focus lies on the Knowledge Meta Process which includes the engineering of ontologies as part of knowledge management applications. We offer a detailed description of steps necessary to develop and employ ontologies.

2. **Tool support:** We implemented numerous specialized tools that support the ontology engineering steps of the Knowledge Meta Process. Our tools fulfill major requirements that rise in the area of ontology engineering, *viz.* support for (i) ontology languages, (ii) flexible extensibility, (iii) methodology support, (iv) collaboration, and (v) inferencing. The tools were realized as extensions to the already existing ontology engineering environment OntoEdit, which provides basic functionalities for ontology engineering.

   Taken together, the On-To-Knowledge Methodology and its accompanying tool support provide an advanced framework for engineering ontology based applications that is highly re-usable for different kinds of domains.

3. **Case studies:** We applied the On-To-Knowledge Methodology and the tools in several case studies, *viz.* the three different knowledge management case studies of the On-To-Knowledge project and the development of the portal for the OntoWeb thematic network. All case studies provide real world experiences and show different aspects of ontology based knowledge management applications.

   The scenarios and implementations range

   (i) from **corporate intranets** to **the Web**,
   (ii) from **industry** to **academia**, and
   (iii) from **prototypes** to **productive systems**.

   Thus, we present the application of the On-To-Knowledge Methodology and its accompanying tool support in a large spectrum of practical scenarios for ontology based knowledge management applications.

   Our approach has already been successfully taken up by other projects, as we will show at the end of the work in the outlook.

To illustrate the tool support for the On-To-Knowledge Methodology, we develop step by step a comprehensive **example ontology** about the topic of this thesis in Part III.

1.3 Reader's Guide

Overview

To help with the reading of this work, every chapter is preceded with a brief introductory overview that is accompanied by a small icon like this paragraph. Each overview explains how the chapter is structured and how the work presented fits in the overall structure.
References: If existent, we will give references to existing publications that form the basis for a chapter.

Part I provides the **Foundations** for this thesis. In this chapter the contributions of this thesis are motivated and introduced. "Knowledge" and how knowledge can be transferred is approached before the building blocks of knowledge management (KM) are described to show the diversity of relevant aspects for applying KM. The concept "ontology" in the scope of this work is illustrated. Current trends of using such ontologies for knowledge management, coined in the vision of the Semantic Web, motivate the need for advanced methodologies, tools and case studies as presented in the following parts. At the end we present a working definition for ontologies that covers the scope of this work.

Part II introduces the **On-To-Knowledge Methodology** as the methodological background for this work. The methodology is a process-driven approach for developing, deploying and using ontology based knowledge management applications. The methodology consists of two main processes: the "Knowledge Meta Process" (KMP) for developing and deploying such applications and the "Knowledge Process" (KP) for using them. Ontology engineering is a major part of the KMP.

Part III desribes **Tool Support** for supporting the application of the On-To-Knowledge Methodology. It starts by motivating and identifying main requirements for tools that support ontology engineering, *viz.* ontology engineering environments (OEE). Following up, we introduce OntoEdit as an instance of such OEEs and subsequently show how OntoEdit meets the introduced requirements: (i) support for ontology languages, (ii) flexible extensibility, (iii) methodology support, (iv) support for collaboration between domain experts and/or ontology engineers, and (v) usage of inferencing capabilities. This work mainly contributed to fulfilling the last three requirements. We will present in detail the specialized tool support for steps of the Knowledge Meta Process.

Part IV illustrates in several **Case Studies** how the methodology and tools presented before are applied in different scenarios. On the one hand, three case studies of the On-To-Knowledge project represent a broad spectrum of use cases in corporate intranets. Firstly, there are three industry sectors involved: insurance, telecom and energy. Secondly, the partners come from three countries with different cultures. Thirdly, they are facing various aspects of knowledge management problems, (i) "Skills Management", (ii) "Communities of Knowledge Sharing" and (iii) "Virtual Organization". On the other hand, the Semantic Portal of the thematic network OntoWeb illustrates the application of the presented work in the World Wide Web. All case studies provide real world experiences and show different aspects of ontology based knowledge management applications.

Part V depicts **Related Work** on methodologies, tools and case studies and summarizes the main contributions in **Conclusions** before it ends with a presentation of already visible impacts of this work and future outlook.

Part VI contains additional material in an **Appendix**, *e.g.* more detailed illustrations of OntoEdit's technical features.

2 Knowledge Management

Overview

This chapter begins by approaching the concept "Knowledge" and how knowledge can be transferred in Section 2.1. The next Section 2.2 describes the building blocks of knowledge management (KM) to show the diversity of relevant aspects for applying KM. In Sections 2.3 and Section 2.4 we argue for a shift from document oriented KM towards a knowledge item oriented point of view and introduce two knowledge processes that are the backbone of the methodology presented in Part II.

References: The last two parts of this chapter are mainly based on (Staab et al., 2001).

2.1 Knowledge Transfer

What is knowledge? Davenport and Prusak suggest the following working definition:

> *"Knowledge is a fluid mix of framed experience, values, contextual information, expert insight and grounded intuition that provides an environment and framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organizations, it often becomes embedded not only in documents or repositories but also in organizational routines, processes, practices and norms."*
> (Davenport & Prusak, 1998)

Thus, knowledge can be incorporated by individuals or organizations. Polanyi adds another dimension, the concept of *tacit knowledge*:

> "[...] *that we know more than we know how to say* [...]"
> (Polanyi, 1958; 1974)

Extending this point of view, knowledge can be *implicit*, *i.e.* tacit, or *explicit*, as introduced by Nonaka and Takeuchi (*cf.* (Nonaka & Takeuchi, 1995)). Sharing knowledge requires considering different dimensions of knowledge transfer. Nonaka and Takeuchi stress that, on the on hand, implicit knowledge is difficult to communicate and to formalize, it is "stored in the heads of persons" and therefore known as *embodied knowledge*. On the other hand, explicit knowledge is easily to communicate, it can be formalized on different levels, it can be stored on different media, *e.g.* in documents or databases, and is therefore called *disembodied knowledge*. A core concern for knowledge management is to support the transformation of knowledge in different processes as shown in Table 2.1 (*cf.* (Nonaka & Takeuchi, 1995)).

Table 2.1: Dimensions of knowledge transfer

		To	
		Implicit knowledge	Explicit knowledge
From	Implicit knowledge	**Socialization**	**Externalization**
	Explicit knowledge	**Internalization**	**Combination**

We will briefly explain the four processes for transferring knowledge:

- Implicit knowledge is transferred directly between persons by **socialization**, *i.e.* through one's own receptions, practical experiences and, most important, building of own internal mental models.
- Implicit knowledge can be made explicit, *e.g.* formalized, by **externalization**, *i.e.* articulation of implicit knowledge by using explicit concepts, metaphors and models.
- Explicit knowledge is embodied again by persons through **internalization**. *E.g. learning by doing* supports the creation or extension of internal mental models of persons.
- Explicit knowledge is enriched through **combination**, *i.e.* by systemizing, categorizing, ordering and combining concepts.

2.2 Building Blocks of 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 thorough and systematic management of knowledge – within an enterprise and between several cooperating enterprises. Obviously, a holistic KM approach 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. This is *e.g.* reflected by KM conferences that address numerous aspects of KM (*cf.* (Schnurr et al., 2001; Reimer et al., 2003)).

We will now briefly introduce the main building blocks of knowledge management, *viz.* culture, organization, technology and people (*cf.* (Albrecht, 1993; Schneider, 1996a)).

Culture

Successful enterprises develop their own culture, that guides their thinking and behaving (*cf.* (Nonaka & Takeuchi, 1995)). Enterprise culture includes values, norms, social contexts, knowledge and skills which are shared and accepted by most members of the organization (*cf.* (Schnyder, 1989)). Probst et al. (*cf.* (Probst et al., 1998; 1999)) emphasizes that internal barriers hindering knowledge management to be effective are often grounded in the culture of enterprises, *e.g.* if a request for knowledge from external sources like other departments is seen as incompetency of employees. However, changing culture is a rather expensive and time consuming task. But in the end the reason for success or failure of introducing KM might be the culture of an enterprise.

Organization

The organization of enterprises consists of its internal processes and structures. Since knowledge is now seen as basic economic resource (*cf.* (Drucker, 1993)), knowledge intensive processes, *e.g.* research & development and education, gain ever more attention. They must be supported and continuously improved in order to achieve an improvement in productivity (*cf.* (Davenport et al., 1996)).

Knowledge management needs similar to traditional organizational units as accounting or human resource a well-defined place in the organizational structure with its own budgets, responsibilities and rights (*cf.* (Davenport et al., 1996)). Many companies created new jobs dedicated to this task such as Chief Information Officers (CIO), Chief Knowledge Managers (CKM) and Chief Knowledge Officers (CKO).

While CIOs are mainly responsible for IT strategy, development of systems and general IT management, the CKMs and CKOs are truly responsible for making knowledge management systems and processes an integral part of the daily work. Typically they have, among others, the following organizational and technological responsibilities (*cf.* (Tiwana, 2000)):

- Organizational responsibilities
  - Identifying knowledge gaps
  - Creating a culture of knowledge sharing
  - Creating appropriate metrics
  - Developing communities of practice
  - Diffusing best practices
  - Training and Education
  - Making knowledge management a part and parcel of routine work
- Technological responsibilities
  - Building directories (*e.g.* skills and knowledge directories)
  - Creating channels for exchange of documents and other codified forms of explicated knowledge
  - Supporting group work
  - Providing tools for collaborative problem solving
  - Building repositories
  - Enabling tacit knowledge transfer

A recent survey took a snapshot of 41 currently practicing knowledge managers via self-reported questionnaires. It summarizes their roles in enterprises, their goals and their self-understanding:

> *"The role of knowledge manager in organizations is a new and growing phenomenon. Our study suggests that knowledge managers are well-educated and experienced individuals who are generally satisfied with their position and the freedom and latitude it affords. The primary goal is to guide their organization towards an understanding of knowledge as an organizational asset so that it can be managed for maximal benefit. As they see it, their key challenge is changing people's behavior. Despite considerable support from top management, they have little direct authority over employees so their levers for effecting change are negotiation, persuasion, and communication."*
> (McKeen & Staples, 2003)

Technology

The two main drivers of practical knowledge management are technology and people, as pointed out by Davenport:

> *Effective management of knowledge requires hybrid solutions of people and technology.*
> (Davenport, 1996)

IT-supported KM applications are typically built around some kind of corporate memory or organizational memory (*cf.* (Kuehn & Abecker, 1997; Abecker et al., 1998; Dieng-Kuntz & Matta, 2002)). Organizational memories integrate informal, semi-formal and formal knowledge in order to facilitate its access, sharing and re-use by members of the organization(s) for solving their individual or collective tasks (*cf.* (Dieng et al., 1999)), *e.g.* as part of business processes (*cf.* (Staab & Schnurr, 2000; 2002)). 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. *I.e.* implicit knowledge of employees is made explicit by means of externalization and stored in the organizational memory. Then it can be accessed by other persons by means of internalization.

Ontologies (*cf.* Chapter 3) 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 (*cf.* (O'Leary, 1998; Gruber, 1995)).

People

The embodied knowledge of people and their abilities to access, integrate and use knowledge are essential for generating value for their enterprises. Their willingness to share their knowledge makes knowledge management possible.

But, as mentioned before, knowledge management is a process in which 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".

We will now focus on relevant processes for introducing KM applications into enterprises.

2.3 Knowledge Processes

There exist various proposals for methodologies that support the systematic introduction of KM applications 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 (*cf.* (Schreiber et al., 1999)).

Typically, these methodologies conflate two processes that should be kept separate in order to achieve a clear identification of issues: whereas the first process addresses aspects of introducing a new KM application 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.

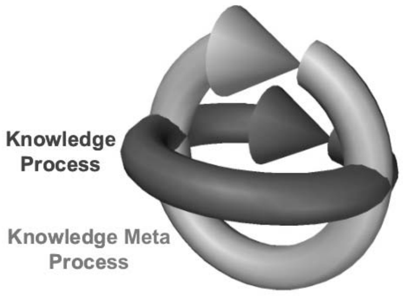


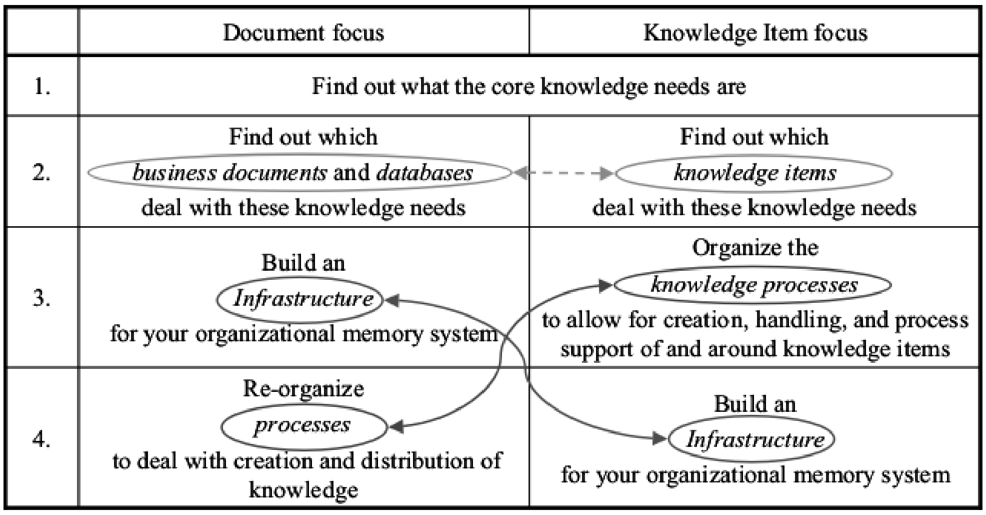
Figure 2.1: Two orthogonal processes with feedback loops

In the Chapters 4 and 5 we will introduce our methodology called On-To-Knowledge Methodology that is based on the two knowledge processes.

2.4 Knowledge Items & Meta Data

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,

Table 2.2: 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

KM applications have tended to serve *either* the needs of easy access to documents (*e.g.* building on groupware, database management systems *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 2.2. 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 (the steps will be described in Chapter 5, *cf.*, *e.g.*, Figure 5.1). 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 Chapter 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 2.2, 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 2.2. 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 help desk 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. In the next chapter we will now focus on "Ontologies", explain what they are and for what purposes they can be used.

3 Ontologies & The Semantic Web

Overview

This chapter provides a basic introduction to ontologies and the Semantic Web. The history and foundations of concept "ontology" are described in Section 3.1. Current trends of using such ontologies for knowledge management, coined in the vision of the Semantic Web, motivate the need for advanced methodologies, tools and case studies. We provide a classification schema for ontologies in Section 3.2. An illustrative example in Section 3.3 shows the core usage of ontologies, *viz.* to support communication.

3.1 History & Foundations

Philosophical Roots

The term "Ontology" (Greek. on = *being*, logos = *to reason*) in its original sense is a philosophical discipline, a branch of philosophy that deals with the nature and the organization of being. Though "Ontology" was first coined in the 17th century, the term is synonymous with "metaphysics" or "first philosophy" as defined by Aristotle during the 4th century BC in his work "Metaphysics, IV, 1". In the context of research on "Ontology", philosophers try to answer fundamental questions like "what is being?" and "what are the features common to all beings?".

"Ontology" vs. "ontology"

According to (Guarino, 1998a) 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* [...]"
> (Guarino, 1998a)

Such a system does not depend on a particular language in the philosophical point of view.

Ontologies in Computer Science

In recent years ontologies have become a topic of interest in computer science (*cf. e.g.* (Gruber, 1995; Uschold & Grueninger, 1996; Noy & Hafner, 1997; Maedche & Staab, 2001; Fensel, 2001)). There are different 'definitions' in the literature of what an ontology should be. Some of them are discussed in (Guarino, 1997; van Heijst, 1995), the most prominent being published by Tom Gruber (*cf.* (Gruber, 1993; 1995)) during his efforts for the *Knowledge Sharing Effort* (KSE, (Neches et al., 1991)):

> *"An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what 'exists' is that which can be represented."*
> (Gruber, 1993)

A *conceptualization* refers to an abstract model of some phenomenon in the world by identifying the relevant concept of that phenomenon. *Explicit* means that the types of concepts used and the constraints on their use are explicitly defined. This definition is often extended by three additional conditions: "An ontology is an explicit, *formal* specification of a *shared* conceptualization *of a domain of interest*". *Formal* refers to the fact that the ontology should be machine readable (which excludes for instance natural language). *Shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted as a group. The reference to *a domain of interest* indicates that for domain ontologies one is not interested in modeling the whole world, but rather in modeling just the parts which are relevant to the task at hand (we will present a classification for different types of ontologies in Section 3.2).

Gruber also introduces as part of the "Frame Ontology" kinds of components, which are used to model ontologies (*cf.* (Gruber, 1993)), *e.g.*: **classes** (often also called **concepts**), **relations** (often, especially in the RDF community, also called **properties**), **axioms**, and **instances**[1]. Concepts are organized in taxonomies through which inheritance mechanisms can be applied. Relations represent a type of interaction between

[1]Readers that are interested in more formal definitions are referred to *The Karlsruhe Perspective on Ontologies*, *cf.* (Bozsak et al., 2002), that was developed in a joint effort of members of the Institute AIFB.

concepts of the domain. Axioms are used to model sentences that are always true. They are also used to represent knowledge that cannot be formally defined by the other components. And, last but not least, instances can be instances of concepts and of relations.

As shown above, an important aspect of ontologies is their consensual character. As mentioned previously, people (*cf.* Section 1.1) need to speak a common language to share knowledge (*cf.* (Davenport & Prusak, 1998)). Users of an ontology first need to agree on a shared conceptualization to enable knowledge sharing on a conceptual level, the so-called *knowledge level* (*cf.* (Newell, 1982)). The same holds for software agents. Considering the fact that different software agents might use different ontologies, they need to be able to reach consensus on the ontology used for communication:

> *"Software agents are communicating with each other via messages that contain expressions formulated in terms of an ontology (ontology driven communication). In order for a software agent to understand the meaning of these expressions, the agent needs access to the ontology they commit to."*
> (Guarino, 1998a)

This is reflected in the *FIPA-ACL*, the *Agent Communication Language* specified by the FIPA[2]. In the specification a special parameter is reserved to reference the used ontology for a message exchanged by agents. Some implementations already make use of explicit ontologies (*cf.*, *e.g.*, (Smolle & Sure, 2002)).

Typically 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. Often a form of First-Order Logic (FOL) theory is used to represent these assumptions. Vocabulary is defined as unary and binary predicates, called concepts and relations, respectively.

The use of logic as a basis for knowledge representation languages was first proposed by McCarthy in 1959 (McCarthy, 1959). His intention was to model knowledge instead of hard-wiring it into specific programs to make it more modular and therefore more reusable:

> *"Expressing information in declarative sentences is far more modular than expressing it in segments of computer programs or in tables. Sentences can be true in a much wider context than specific programs can be used. The supplier of a fact does not have to understand much about how the receiver functions or how or whether the receiver will use it. The same fact can*

[2]FIPA is the official abbreviation for "Foundation for Intelligent Physical Agents", see `http://www.fipa.org/`

be used for many purposes, because the logical consequence of collections of facts can be available."
(McCarthy, 1989)

Ontologies have been successfully applied in numerous scenarios including (taken from (Erdmann, 2001)):

- Natural Language Processing and Machine Translation (*e.g.* (Miller, 1995; Staab et al., 1999; Dahlgren, 1995)),
- Knowledge Engineering (*esp.* Problem Solving Methods (Fensel, 2000)),
- Knowledge Management (*cf.* (Abecker et al., 1997; Staab & Schnurr, 2000; Sure et al., 2000)),
- Engineering Disciplines (*e.g.* (Borst & Akkermans, 1997; Pocsai, 2000)),
- Electronic Commerce (*e.g.* RosettaNet[3] and Ontology.org[4]),
- Information Retrieval and Information Integration (*e.g.* (Kashyap, 1999; Mena et al., 1998; Wiederhold, 1992)),
- Web Catalogs (*e.g.* Yahoo! (Labrou & Finin, 1999)),
- Intelligent Search Engines (*e.g.* Ontobroker (Decker et al., 1999), SHOE (Heflin & Hendler, 2000), Getess (Staab et al., 1999), OntoSeek (Guarino et al., 1999)),
- Digital Libraries (*e.g.* (Amann & Fundulaki, 1999)),
- Enhanced User Interfaces (*e.g.* (Kesseler, 1996), Inxight[5]),
- Software Agents (*e.g.* (Gluschko et al., 1999; Smith & Poulter, 1999)) or
- Business Process Modelling (*e.g.* (Decker et al., 1997; TOVE, 1995; Uschold et al., 1998)).

Further examples can be found *e.g.* in (Sowa, 2000; Guarino, 1998a; Noy & Hafner, 1997; Jasper & Uschold, 1999; Studer et al., 1998; Sure & Schnurr, 2003). However, these have been mostly academic applications. With the rise of the Semantic Web, ontologies gain ever more attention and industry is beginning to take up the first results and to offer commercial products (*e.g.* (Moench, 2003)). *E.g.* one of the primary goals of the OntoWeb[6] thematic network is the following one:

[3]RosettaNet, http://RosettaNet.org/
[4]Ontology.Org, http://www.Ontology.org/
[5]Inxight Inc., http://www.inxight.com/
[6]OntoWeb.org, http://www.ontoweb.org/

> *"Demonstrating to industry how ontologies can be applied to particular problems in Knowledge Management, Electronic Commerce, and Enterprise Integration, and identifying problems in industry that can be addressed in scientific research."*
> (OntoWeb, 2001)

The network started in Summer 2001 and has currently over 100 members coming from academia and industry that share an interest in ontologies and the Semantic Web. In Chapter 12 we will describe more detailed the goals of OntoWeb and in particular the contribution of this work with respect to the "SEmantic portAL" of the network.

The vision of the Semantic Web is described as

> *"... an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications ... data can be shared and processed by automated tools as well as by people."*
> (W3C, 2001)

As such, it offers a large potential for supporting issues like "integration" and "re-use", two of the main practical problems of computer science. Quite naturally, with the growing interest on the Semantic Web the number of ontologies for various domains will rise. Although many researchers are working now in the research area Semantic Web on different problems, there is still a lack of work that integrates aspects to give a clear picture and overview of how to develop and use ontology based applications for practical knowledge management problems.

This thesis is motivated by the urgent need for comprehensive methodologies, that (i) guide domain-independently the introduction and running of ontology-based applications, (ii) have appropriate tool support and (iii) are accompanied by illustrative show cases. Thus, the thesis *e.g.* contributes to the goals of the OntoWeb network as described above.

3.2 Classification of Ontologies

Different classification systems for ontologies have been developed (van Heijst, 1995; Guarino, 1998a; Jasper & Uschold, 1999). A classification system that uses the subject of conceptualization as a main criterion has been introduced by Guarino (*cf.* (Guarino,

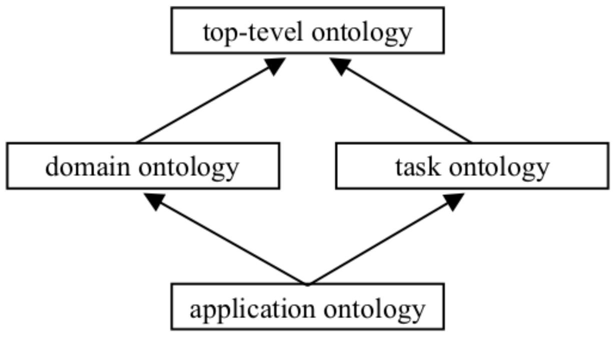


Figure 3.1: Different kinds of ontologies and their relationships

1998a)). He suggests to develop different kinds of ontologies according to their level of generality as shown in Figure 3.1.

Thus, 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. Such unified top-level ontologies aim at serving large communities of users and applications. They facilitate the (semi-)automatic integration and combination of different ontologies that are mapped to the same top-level ontology. Prominent examples of such ontologies include the top-level ontology by John Sowa (*cf.* (Sowa, 2000)), the IEEE SUO[7] and DOLCE (*cf.* (Gangemi et al., 2002a)). Recently, these kinds of ontologies have been also introduced under the name **foundational ontologies** and efforts exist on creating a modularized "Foundational Ontologies Library" (*cf.* (Gangemi et al., 2002a)).
- **Domain ontologies** describe the vocabulary related to a specific domain (such as vines or cars), *e.g.* by specializing concepts introduced in a top-level ontology.
- **Task ontologies** describe the vocabulary related to a generic task or activity (such as drinking or selling), *e.g.* by specializing concepts introduced in a top-level ontology.
- **Application ontologies** are the most specific ontologies. Concepts in application ontologies often correspond to roles played by domain entities while performing a certain activity, *i.e.* application ontologies are a specialization of domain and task ontologies. They form a base for implementing applications with a concrete domain and scope.

[7]Standard Upper Ontology, http://suo.ieee.org

Noteworthy are also **common sense ontologies** (*cf.* (Pirlein, 1995; Lenat & Guha, 1990; Lenat, 1995)) which are used to model aspects relevant for many other applications. As such, they contain typically concepts of space and time and are intended to be highly re-usable.

In this work we refer by ontology to domain and in particular to application ontologies.

3.3 Communication with Ontologies

Reference & Meaning

The *meaning triangle* (*cf.* (Odgen & Richards, 1923), in the tradition of Frege, *cf.*, *e.g.*, (Frege, 1892)) is used to define the interaction between symbols or words, concepts and things of the world (*cf.* Figure 3.2). 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 the concept and the appropriate thing in the world (= object). We will now illustrate some practical aspects of the meaning triangle by an example.

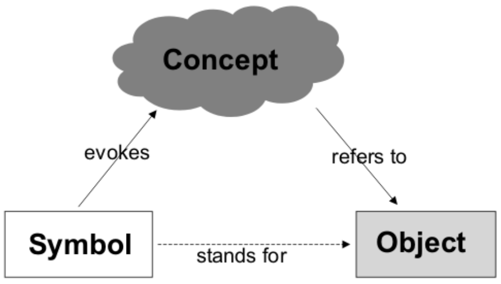


Figure 3.2: The Meaning Triangle

Communication Example

The following example and figure has been taken from (Maedche, 2002a), but was slightly modified to fit the the skills management case study in Chapter 11. Figure 3.3 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. Secondly, we deal with symbols and syntactic structures that are exchanged. Thirdly, we analyze models with their specific semantic structures that correspond to certain domains, *e.g.* geography or skills.

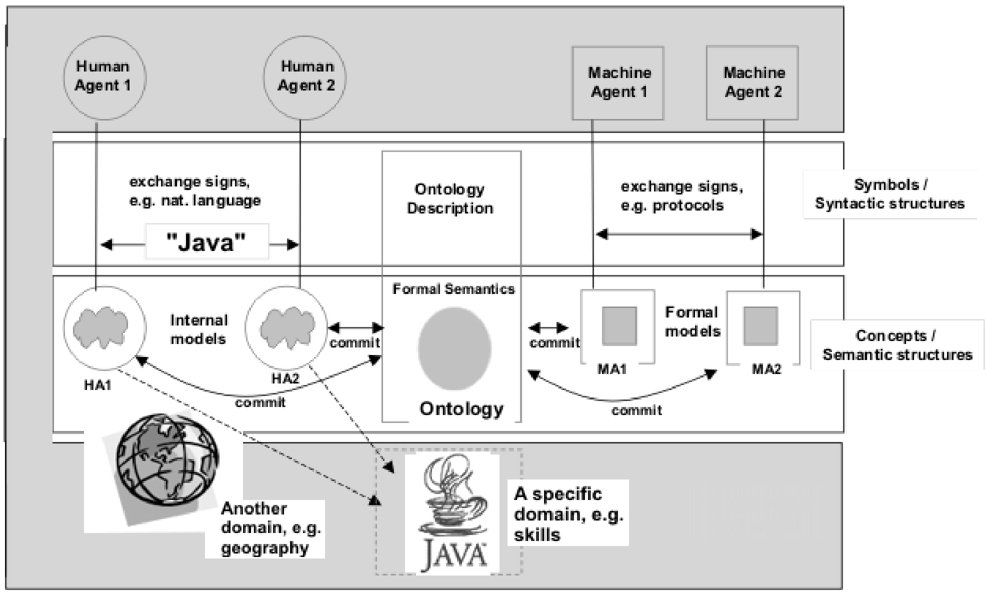

Human Agent 1
Human Agent 2
Machine Agent 1
Machine Agent 2
exchange signs, e.g. nat. language
"Java"
Ontology Description
exchange signs, e.g. protocols
Symbols / Syntactic structures
Formal Semantics
Internal models
commit
HA1
HA2
commit
Ontology
commit
Formal models
MA1
MA2
commit
Concepts / Semantic structures
Another domain, e.g. geography
A specific domain, e.g. skills
JAVA

Figure 3.3: Communication between human and/or software agents

Let us first consider the left side of Figure 3.3 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.

Part II

On-To-Knowledge Methodology

"If you can't describe what you are doing as a process,
you don't know what you're doing."
— W. Edwards Deming

4 Knowledge Meta Process

This part introduces the **On-To-Knowledge Methodology** for developing, deploying and using ontology based knowledge management applications. The methodology consists of two main processes: the “Knowledge Meta Process” (KMP) for developing and deploying such applications and the “Knowledge Process” (KP) for using them.

Overview

This chapter is about the KMP, the following Chapter 5 is about the KP. Section 4.1 introduces relevant processes for developing and deploying knowledge management applications. We focus on the KMP as a core process and introduce in Section 4.2 an overview of the KMP steps. Each step is subsequently described in the following sections, *viz.* Feasibility Study in Section 4.3, Kickoff in Section 4.4, Refinement in Section 4.5, Evaluation in Section 4.6 and Application & Evolution in Section 4.7.

References: This chapter is mainly based on (Staab et al., 2001), (Sure et al., 2002c) and (Sure & Studer, 2002a).

4.1 Developing and Deploying Knowledge Management Applications

To implement and invent any KM application, one has to consider different processes (*cf.* Figure 4.1). We experienced in the case studies presented in Part IV mainly three interacting processes that influenced the projects, *i.e.* “Knowledge Meta Process”, “Human Issues” and “Software Engineering”.

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 KM applications – strongly depends on the acceptance by the involved people. As a consequence, “quick wins” are recommended for the initial phase

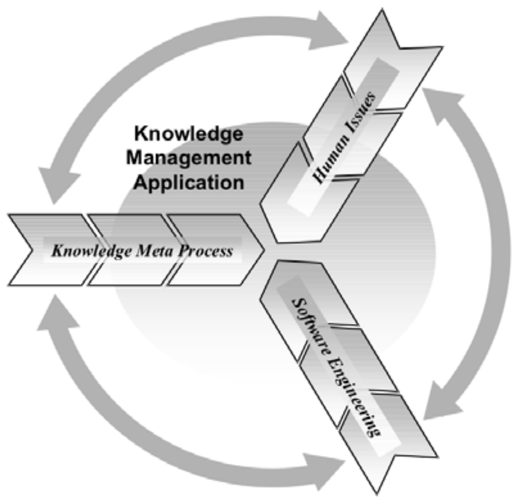


Figure 4.1: Relevant processes for developing and deploying KM applications

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) for knowledge management applications has to fit to the other processes. The now presented On-To-Knowledge Methodology is cyclic and includes iterative steps, ideally the software engineering methods should also explicitly include cycles and iterations. A prominent examples for such a methodology is (Jacobson et al., 1999). Although the On-To-Knowledge Methodology inherits the notion of the core steps from object-oriented programming (*cf.* (Jacobson, 1998)), it does actually not require or rely on a particular SE methodology.

As introduced in Section 2.3, we distinguish two main processes. In the following sections we will first focus on the Knowledge Meta Process as the core process and illustrate some cross-links to the other mentioned processes. The Knowledge Process is subsequently presented in the next Chapter 5. The methodology described in this part was (mainly) developed and applied in the On-To-Knowledge[1] project (*cf.* Chapter 11).

4.2 Steps of the Knowledge Meta Process

The Knowledge Meta Process (*cf.* Figure 4.2) describes the development and employment of ontology based KM applications. The Knowledge Meta Process consists of five

[1]EU IST-1999-10132 On-To-Knowledge, see `http://www.ontoknowledge.org/`

main steps. Each step has numerous sub-steps, requires a main decision to be taken at the end and results in a specific outcome:

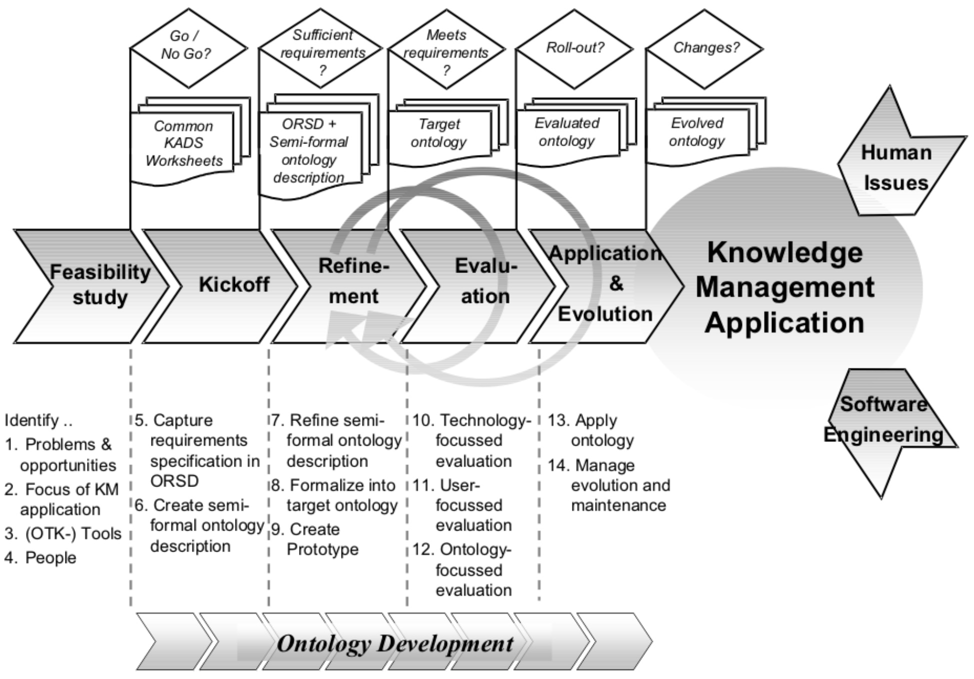

Go / No Go?
Sufficient requirements ?
Meets requirements ?
Roll-out?
Changes?
Common KADS Worksheets
ORSD + Semi-formal ontology description
Target ontology
Evaluated ontology
Evolved ontology
Human Issues
Feasibility study
Kickoff
Refine-ment
Evalu-ation
Application & Evolution
Knowledge Management Application
Software Engineering
Identify ..
1. Problems & opportunities
2. Focus of KM application
3. (OTK-) Tools
4. People
5. Capture requirements specification in ORSD
6. Create semi-formal ontology description
7. Refine semi-formal ontology description
8. Formalize into target ontology
9. Create Prototype
10. Technology-focussed evaluation
11. User-focussed evaluation
12. Ontology-focussed evaluation
13. Apply ontology
14. Manage evolution and maintenance
Ontology Development

Figure 4.2: The Knowledge Meta Process

- 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 box depicting a phase the most important sub-steps are listed, *e.g.* "Refinement" consists of the sub-steps "Refine the semi-formal ontology description", "Formalize into target ontology" and "Create Prototype". *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. Typically the major outcomes serve as a base for the decisions to be taken.
- As indicated, some steps need to be performed in iterative cycles.

A main part of the Knowledge Meta Process is covered by ontology engineering, beginning with the “Kickoff” phase. In Part III we will describe dedicated tool support for each of the steps during ontology development. In Part IV we will describe lessons learned during the application of the On-To-Knowledge Methodology in different case studies.

4.3 Feasibility Study

Any knowledge management system may function properly only if it is seamlessly integrated in the organization in which it is operational. Many factors other than technology determine success or failure of such a system (*cf.* Section 4.1). To analyze these factors, we initially start with a ***feasibility study***, *e.g.* to identify problem/opportunity areas and potential solutions. In general, a feasibility study serves as a decision support for economical, technical and project feasibility, determining the most promising focus area and target solution.

CommonKADS Worksheets

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:
  1. Identifying problem/opportunity areas and potential solutions.
  2. 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:
  1. 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.
  2. 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 4.3. Building the organization, task and agent model is done by following a series of steps supported by practical and easy-to-use worksheets and checklists (we provide a brief summary of the most relevant steps and refer for a detailed description of these steps to the CommonKADS methodology (Schreiber et al., 1999)).

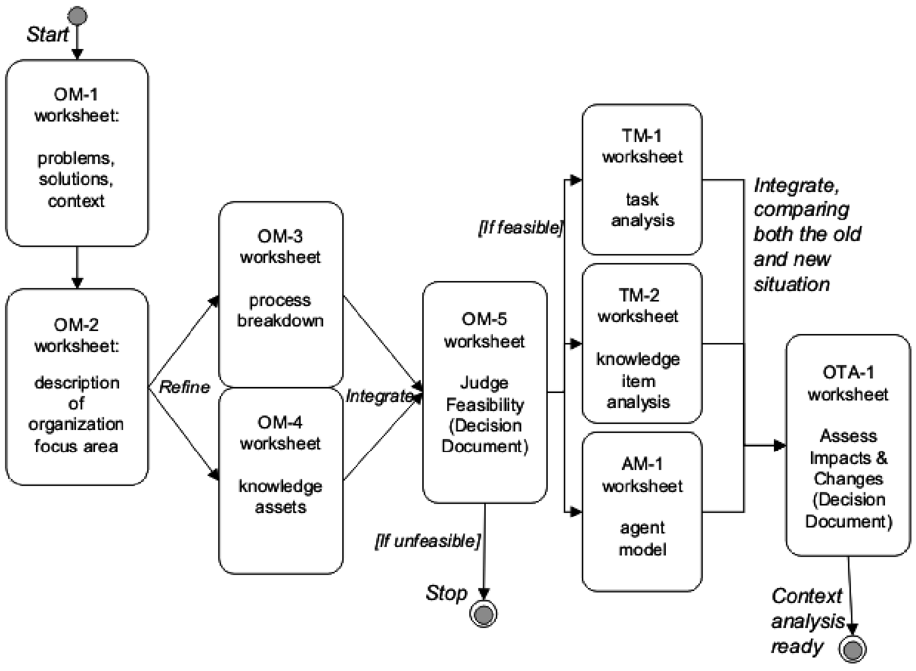

Start
OM-1 worksheet: problems, solutions, context
OM-2 worksheet: description of organization focus area
Refine
OM-3 worksheet process breakdown
OM-4 worksheet knowledge assets
Integrate
OM-5 worksheet Judge Feasibility (Decision Document)
[If feasible]
[If unfeasible]
Stop
TM-1 worksheet task analysis
TM-2 worksheet knowledge item analysis
AM-1 worksheet agent model
Integrate, comparing both the old and new situation
OTA-1 worksheet Assess Impacts & Changes (Decision Document)
Context analysis ready

Figure 4.3: CommonKADS worksheets for feasibility study

The organization model focuses on problems and opportunities. These issues are discussed in the worksheets OM-1 – OM-4 that are described in the following.

- Worksheet OM-1 explains the problems, solutions and the organizational context, *e.g.* including (i) the mission, vision and goals of the organization, (ii) important external factors the organization has to deal with, (iii) strategy of the organization and (iv) its value chain and the major value driver.
- Worksheet OM-2 concentrates upon the more specific aspects of an organization. This includes *e.g.* how the business is structured, what staff is involved and what resources are used.
- Worksheet OM-3 breaks down the business process into smaller tasks, because an envisaged KM application always carries out specific tasks and this has to fit into the process as a whole.

- Worksheet OM-4 describes the knowledge assets of an organization. It provides *e.g.* detailed information on who possesses a knowledge asset and in which tasks it is used.
- Worksheet OM-5 finally provides a checklist for the feasibility decision. All key implications are wrapped up to support the decision-making process. Key questions in the process are:
  - What is the most promising opportunity area for applications, and what is the best solution direction?
  - What are the benefits versus the costs (business feasibility)?
  - Are the needed technologies for this solution available and within reach (technical feasibility)?
  - What further projet actions can successfully be undertaken (project feasibility)?

If a project is considered feasible, the task and the agent model help to prepare the start of the ontology development in the kickoff phase by zooming in on the features of the relevant tasks, the agents that carry them out, and on the knowledge items used by the agents in performing tasks.

- Worksheet TM-1 can be seen as a refinement of OM-3. All identified tasks are now being described in detail, *e.g.* covering goal and value of tasks, their timing and competencies needed for successful task performance.
- Worksheet TM-2 is a refinement of OM-4 and provides a knowledge bottleneck analysis to uncover the areas for improvement.
- Worksheet AM-1 adds to the task oriented point of view the perspective of individual agents (*e.g.* knowledge workers). The purpose of the agent model is to understand the roles and competencies that various actors in the organization bring with them to perform a (shared) task.
- Worksheet OTA-1 integrates the task and agent model into a checklist that provides a detailed insight into the impact of a KM application, and especially what improvement actions are possible or necessary in the organization in conjunction with the introduction of such an application.

Outcome & Decision

In the end the two decision documents OM-5 and OTA-1 serve as a base for (i) deciding whether the project *per se* is started, and (ii) identifying the problems & opportunities, the focus of the application, the technology and in particular the tools needed that are available and, last but not least, the people involved.

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.

4.4 Kickoff

Capture Requirements Specification

In the Kickoff phase the actual development of the ontology begins. Similar to software engineering and as proposed by (Fernández-López 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 next step, the gathering of the semi-formal description of the ontology. The ORSD should also guide ontology engineers in the refinement phase 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[2]. In detail, the ORSD contains the following information.

ORSD, part (i): Goal, Domain and Scope

In the beginning one should specify the particular domain in use, which might *e.g.* 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 from the Swiss Life case study (*cf.* Section 11.4): "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".

[2]The reuse of ontologies, *e.g.* by merging or aligning them, is not covered by this methodology. We describe the process of developing an ontology from scratch. However, the merging and alignment can be integrated in the methodology as part of the kickoff and refinement steps. For existing methodologies and tools dedicated to merging and aligning we refer to (Russ et al., 1999; Noy & Musen, 2000; McGuinness et al., 2000a; Stumme & Maedche, 2001; Pinto et al., 2002).

ORSD, part (ii): Design Guidelines

Due to the nature of our related case studies (*cf.* Part IV) we focus on pragmatic design guidelines that help users who are not familiar with modeling.

The guidelines 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 and relations. A typical approach for a naming convention is to begin all concepts with capitals and all relations with small caps. 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 (*e.g.* as the OntoAnalyzer which is presented in Section 9.2). There exist standardized naming conventions for concepts and relations such as (ISO 704, 1987).

(Noy & McGuinness, 2001) proposes a set of **pragmatic guidelines** for modelling ontologies. Especially domain experts not familiar with modelling (*e.g.* at Swiss Life) found these guidelines quite helpful. The guidelines are based on experiences with Protégé which is an ontology editor similar to OntoEdit (*cf.* next Part III on tool support). Therefore the guidelines were easily applicable when using OntoEdit for modelling (we even formalized some of them as constraints that can be automatically checked with the OntoAnalyzer, *cf.* Section 9.2). We now sketch some of the guidelines found in (Noy & McGuinness, 2001) which we found most useful in the case studies.

- *"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 "light-weight" 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 **formal guidelines** exist regarding the proper creation of "clean" taxonomical structures (*cf.*, *e.g.*, (Guarino & Welty, 2000a)). These guidelines are based on philosophical notions and resulted in the development of the OntoClean methodology including guidelines on the **use and misuse of subsumption** in ontology engineering (*cf.*, *e.g.*, (Guarino & Welty, 2002)). The OntoClean methodology aims at the formal evaluation of ontologies, but inherently provides guidelines as mentioned before. We discuss OntoClean and its application in Section 9.3 which is about the implementation of the OntoClean plugin.

A common problem in knowledge engineering is that objects may play different **roles** (*cf.* (Steimann, 2000)) simultaneously, resulting in multiple classifications of objects. Similar to that, objects may play the same role several times, simultaneously. A typical example is that an employee may hold several employments at the same time. The main reason to distinguish multiple occurrences in the same role is that each occurrence of the object in a role is associated with a different state. For example, an employee has one salary and one office address per job.

As (Steimann, 2000) points out, the interest in roles has grown continuously in recent years. But although there appears to be a general awareness that roles are an important modelling concept, until now no consensus has been reached as to how roles should be represented or integrated into the established modelling frameworks. This may partly be due to the different contexts in which roles are introduced, and partly to the different problems one is trying to solve with them. However, the divergence of definitions contradicts the evident generality and ubiquity of the role concept, and hampers its general acceptance as a modelling construct. Thus, there are currently no guidelines that help in modelling roles, but (Steimann, 2000) gives an exhaustive overview of existing definitions in the literature.

ORSD, part (iii): 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 typically available knowledge sources as an example:

- domain experts (interviews, competency questionnaires)
- (reusable) ontologies
- dictionaries
- thesauri
- other 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
- *etc.*

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" (*cf.* Section 12.2). Especially the Swiss Life case study (*cf.* Section 11.4) explored the integration of various existing knowledge sources.

ORSD, part (iv): (Potential) Users and Usage Scenarios

This includes a list of potential users or user groups and a description of each usage scenario. These scenarios might be described by potential users 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.

ORSD, part (v): Supported Applications

This is a draft of the ontology based knowledge management application and its system and software environment, which links the Knowledge Meta Process to software engineering (*cf.* Section 4.1). 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.

ORSD, part (vi): Usage of Competency Questions

The usage scenarios (see above) describe aspects of 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) (*cf.* (Grueninger & Fox, 1994; Uschold & Grueninger, 1996)). They represent a set of possible queries to the system, indicating the scope and content of the application or domain ontology (*cf.* Section 3.2). In Section 7.1 we will show the integration of CQs into the ontology engineering environment OntoEdit, in Section 11.4 we present an example from the Skills Management case study.

Create semi-formal Ontology Description

Similar to the work presented in (Neubert, 1994) we use different formalization levels during the creation of ontologies. Thus, the next step after capturing requirements specifications is the creation of a semi-formal ontology description. This is actually the beginning of the modelling activity. In the case studies we relied on the creation of mind maps™ to capture quickly and intuitively the domain knowledge from domain experts. In Section 7.2 we present the Mind2Onto framework that bridges the gap between the rather weakly structured mind maps™ towards ontologies.

In the next Section 4.5 we will elaborate more on some guidelines for the creation of initial mind maps™. The guidelines help to create them in a way that facilitates the transformation into an ontology.

Outcome & Decision

The outcomes of this phase are (i) the ontology requirement specification document (ORSD)), and (ii) the 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™.

If the requirements are sufficiently captured, one may proceed with the next phase. The decision is typically taken by ontology engineers in collaboration with domain experts. "Sufficiently" in this context means, that from the current perspective there is no need to proceed with drafting domain knowledge, but rather the existing structure needs to be refined in a separate session.

4.5 Refinement

Refine semi-formal ontology description

We will now describe the following issues that are relevant during the refinement of the semi-formal ontology description: (i) top–down *vs.* bottom-up modelling, (ii) the further usage of competency questions, and (iii) guidelines for creating initial mind maps™.

Top–down *vs.* Bottom–up Modelling

During the kickoff and refinement phase one might distinguish in general two concurrent approaches for modeling: 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 (*cf.* Part IV) we encountered more 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, a typical **bottom–up** approach may be applied (*e.g.* in *ontology learning*, *cf.* (Maedche, 2002a)). 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. 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, which 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.

Further Usage of Competency Questions

During the refinement phase the competency questions may be analyzed to derive relevant concepts, relations and instances that should be included into the ontology (as shown detailed in Section 7.1). Typically the semi-formal description serves as a baseline that is refined by concepts and relations found in the competency questions.

Guidelines for creating initial Mind Maps™

Typically, a mind map™ does not provide inherently a consistent taxonomy of concepts, but rather associatively linked elements. Such elements might represent single concepts, multiple concepts, concepts related by a relations *etc.* At first hand, they provide a quick and intuitive way for capturing knowledge from the domain experts, even in the absence of ontology engineers. The effort of transforming mind maps™ into ontologies has currently to be performed manually, typically by ontology engineers. However, we provide valuable tool support (*cf.* Section 7.2) to bridge that gap.

In two of the case studies (*cf.* Sections 11.4 and 11.5) we made good experiences with guiding the domain experts towards an *ontology-like* mind map™, that already provides some structural elements. We relied on the following simple guidelines:

- Try to use only single nouns (whenever possible) for naming elements of a mind map™.
- Try to avoid relationships (whenever possible) in the initial kickoff phase. Add them later in the refinement phase, after an initial taxonomy of concepts has been established.

These guidelines helped to create mind maps™ that are quite easily transformed into basic structures of ontologies. For instance, without much interference of ontology engineers we got in our scenarios “subtopic of” structures that could be easily modelled as a proper ontological structure.

However, a drawback of the mind maps™ is their weakness for modelling (potentially numerous) relationships and to illustrate the inheritance implied by “is-a” structures. Therefore, the initial semi-formal ontology description is much more like a first draft version of a taxonomical structure *without* relations or inheritance. Thus, in the formalization phase required relations and attributes (and possibly instances and axioms) need to be added.

Formalize into Target Ontology

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 – mainly derived out of the competency questions.

The ontology engineer adds different types of relations as analyzed *e.g.* in the competency questions to the taxonomic hierarchy, *e.g.* by using typical ontology engineering environments. In Chapter 6 we will motivate typical requirements for such environments and introduce OntoEdit as an instance of such environments.

However, this step is cyclic in itself, meaning that the ontology engineer now may need to start interviewing domain experts again and to 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 (*cf.*, *e.g.*, (Decker, 2002) for a comprehensive comparison of typical ontology languages). Also, when using modelling tools, they typically are not capable of exporting all current ontology languages, an example for OntoEdit's limitations of exporting DAML+OIL is given in Appendix C.

Create Prototype

As proposed by (Fernández-López et al., 1999), it is very valuable to have "evolving prototypes" of ontologies. Since we typically do not model top-level ontologies, but rather domain or task ontologies (*cf.* Section 3.2), we recommend to build a prototype of the envisaged ontology based application at the end of the refinement phase. This makes the evaluation of ontologies, especially the user-focussed evaluation (*cf.* next section) easier if not possible at all.

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.

Outcome & Decision

The outcome of this phase is the "target ontology" (as mentioned above, potentially embedded into a prototype application), that needs to be evaluated in the next step. In comparison to the semi-formal ontology description, the target ontology needs to fulfill formal ontology definitions (*e.g.* like given in (Bozsak et al., 2002)).

The major decision that needs to be taken to finalize this step is whether the target ontology is mature enough to evaluate the requirements captured in the previous kickoff phase. This decision will be most likely 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.

4.6 Evaluation

To describe the evaluation task, we cite (Gómez-Pérez, 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.*"
> (Gómez-Pérez, 1996)

The requirements specifications and particularly the competency questions that were captured during the kickoff serve therefore as a frame of reference for the evaluation of ontologies. An obvious strategy is therefore to check whether the ontology possesses enough competence to answer all competency questions.

However, considering the fact that we aim at building applications, evaluation is not only restricted to ontologies themselves. Additionally one has also to consider technology-related issues as well as an evaluation from a users' perspective. Therefore we distinguish the following three kinds of evaluation:

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 4.1.

Ontology Properties	Technology Properties
Language conformity (Syntax)	Interoperability (*e.g.* Semantics)
Consistency (Semantics)	Turn around ability
	Scalability (including performance and memory allocation)
	Integration into frameworks (*e.g.* by connectors and interfaces)

Table 4.1: Technology-focussed 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.

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?".

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).

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. This includes whether there exist **connectors** (and **interfaces**) to other tools. This concerns (i) connectors *to* other tools and (ii) connectors *from* other tools to the tool.

This evaluation framework for technology oriented evaluation was *e.g.* taken as a baseline for the workshop on "Evaluation of Ontology based Tools (EON2002)", *cf.* Section 12.3. Within an experiment as part of the workshop, different ontology engineering environments were evaluated.

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 4.4) 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. A recent approach for tracking (and using) usage patterns has *e.g.* been presented in (Stojanovic et al., 2002d). 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 11.6.

Ontology-focussed Evaluation

Beside the above mentioned process oriented and pragmatic evaluation methods, there exist also formal evaluation methodologies for ontologies themselves. Typically, one distinguishes in software engineering between "verification" and "validation" to evaluate the quality of software systems. The two concepts can be characterized as follows (*cf.* (Boehm, 1984)):

- **Verification** is building the system right.
- **Validation** is building the right system.

As (Preece, 2000) points out, these methods can be analogously used in knowledge engineering. (Gómez-Pérez, 2003) focusses on the validation and verification of ontologies with respect to the criteria consistency, completeness, conciseness, expandability and sensitiveness. However, our ontology engineering environment OntoEdit (*cf.* Part III) prevents the incorrect modelling of ontologies that do not fulfill the criteria. Therefore we focus on another well-known methodology for formal evaluations of ontologies.

The OntoClean Methodology (*cf.*, *e.g.*, (Guarino & Welty, 2002)) 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., 2002b)). 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, 2000a)).

By attaching the meta relations to concepts of an ontology, they are classified into categories (Sortal, Non-sortal, Role *etc.*). *E.g.*, a concept that is tagged with "+O +I +R" is called a "Type". OntoClean defines an ideal structure of such classified concepts. By defining constraints one is able to derive automatically modelling errors in an "unclean" taxonomy. In Section 9.3 we explain more detailed the foundations of OntoClean and how we implemented a specialized tool support to operationalize the methodology.

Cyclic Approach

If the evaluation reveals insufficient quality of an ontology or an ontology based application, the ontology needs to be refined again. From practical experiences, the iterative steps can be highly cyclic, *i.e.* each refinement step might require a separate evaluation step. In fact, there exist an approach that recommends the concurrent quality ensurance of an ontology during the complete ontology development lifecycle (*cf.* (Fernández-López et al., 1999)).

Outcome & Decision

The outcome of this phase is an evaluated ontology, possibly embedded into the up-and-running application that is ready for the roll-out into a productive system.

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, *i.e.* the requirements set up in the kickoff phase.

4.7 Application & Evolution

Apply Ontology

The application of ontologies in productive systems, or, more specifically, the usage of ontology based systems, is being described in the following Chapter 5 that illustrates the "Knowledge Process". This steps is actually the meeting point of the Knowledge Meta Process and the Knowledge Process.

Manage Evolution and Maintenance

The evolution of ontologies is primarily an organizational process. There have to be strict rules to the update, insert and delete processes of ontologies (*cf.* (Stojanovic et al., 2002a)). We recommend, that ontology engineers gather changes to the ontology and initiate the switch-over to a new version of the ontology only 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. However, systems can help identifying the needs for evaluation, *e.g.* by "usage mining" (*cf.* (Stojanovic et al., 2002d)) or text mining (*cf.* (Maedche, 2002a)).

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 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.

Cyclic Approach

Each request for a change as part of the evolution typically requires an additional cycle beginning by the refinement phase.

Outcome & Decision

The outcome of an evolution cycle is an evolved ontology, *i.e.* typically another version of it. The storage and management of different versions of ontologies is illustrated in the Section 10.1.

The major decision to be taken is when to initiate another evolution cycle for the ontology. This depends on the restrictions that apply for an organization and the evolution strategy one has implemented as part of the organization.

5 Knowledge Process

Overview

The **On-To-Knowledge Methodology** described in **this part** consists of two main processes: the “Knowledge Meta Process” (KMP) for developing and deploying knowledge management applications and the “Knowledge Process” (KP) for using them.

This chapter is about the KP. Section 5.1 introduces the steps of the KP. Subsequently all introduced steps are described in detail in the following sections, *viz.* Knowledge Creation and Knowledge Import in Section 5.2, Knowledge Capture in Section 5.3, Knowledge Retrieval in Section 5.4, and Knowledge Use in Section 5.5.

References: This chapter is partially based on (Staab et al., 2001) and (Sure & Studer, 2002c).

5.1 Steps of the Knowledge Process

Once a KM application is fully implemented in an organization, knowledge processes essentially circle around the following steps of the “Knowledge Process” (*cf.* Figure 5.1). The process steps reflect the typical steps that knowledge workers perform as part of their daily work.

- *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 (*cf.* Section 2.4) 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;

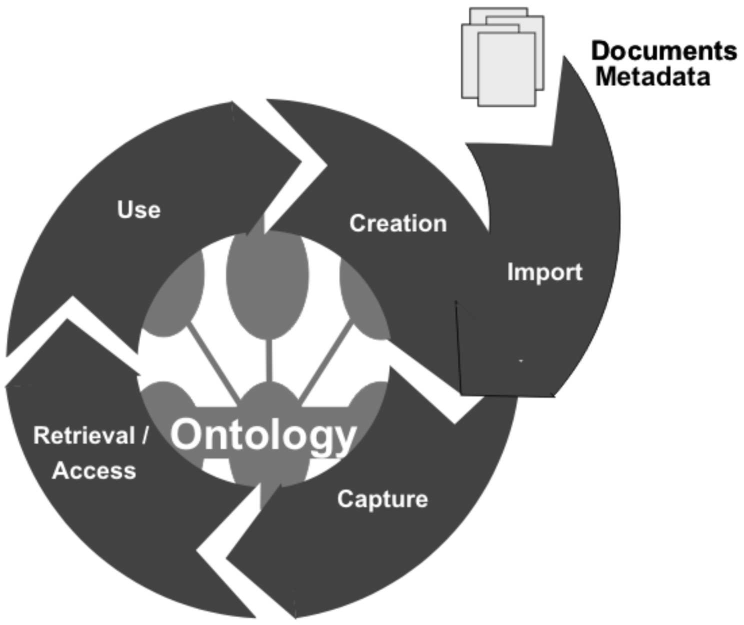


Figure 5.1: The Knowledge Process

- typically, however, the knowledge worker will not only recall knowledge items, but she will process it for further *use* in her context.

Quite naturally, the process is highly cyclic. Knowledge that has been created or imported, captured, retrieved and used typically feeds again into the next cycle.

Ontologies as a backbone for ontology based knowledge management applications lie in the center of the circling processes. In the following sections we will discuss relevant aspects for each step and show ways of using ontologies in the Knowledge Process.

5.2 Knowledge Creation & Import

Creation of knowledge is an integral part of the daily life for knowledge workers. However, the degrees of knowledge that is created may vary from formal to informal knowledge, as we will illustrate in the next subsection. But, at least equally important is the import of external knowledge. We will discuss in the last section the main distinction between creation and import with respect to ontology based applications.

Formal *vs.* Informal Knowledge

The 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 5.1.

Table 5.1: 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

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 (*cf.* (XML/EDI-Group, 2003)) 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 (*cf.* (DC, 2003)) meta data.

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

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 (*cf.* (O'Leary, 1998)) – though it seems implausible that they would contain less useful contents.

Ontology based applications typically make use of meta data (*cf.* Section 2.4). As mentioned before, the meta data conforms to structures provided by ontologies. Noteworthy is the following distinction with respect to the use of ontology based technologies between the import and the creation:

- **Created knowledge** can automatically be enriched with meta data according to an ontology, therefore inherently capturing knowledge contained in the knowledge items. An example implementation is the OntoWeb Semantic Portal. The ontology is used to create forms that automatically enrich created knowledge with meta data according to the underlying ontology. In Section 12.2 we present a detailed example.
- **Imported knowledge** often needs to be enriched in an additional (manual) effort with meta data to make the content accessible. In the next Section 5.3 we will present an approach that supports knowledge capturing, the so-called CREAM framework implemented by the OntoMat–*Annotizer* (*cf.* (Handschuh et al., 2001; 2002)). However, if the external knowledge is already including meta data according to an agreed ontology, the import task becomes fairly easy. This approach is also taken in the OntoWeb Semantic Portal. Partners of OntoWeb who want to contribute knowledge to the portal provide meta data according to the OntoWeb ontology. Thus, the import via syndication can be performed automatically. A detailed description of the architecture can be found in Chapter 12.

5.3 Knowledge Capture

Driven by the knowledge-based industries, knowledge capture gains ever more significance, as outlined in the following citation.

> *"In today's Web-linked and data-rich world, there is a growing need to manage burgeoning amounts of information effectively. Although indexing and linking documents and other information sources is an important step, capturing the knowledge contained within these diverse sources is crucial for the effective use of large information repositories."*
> Introduction to (Gil et al., 2001)

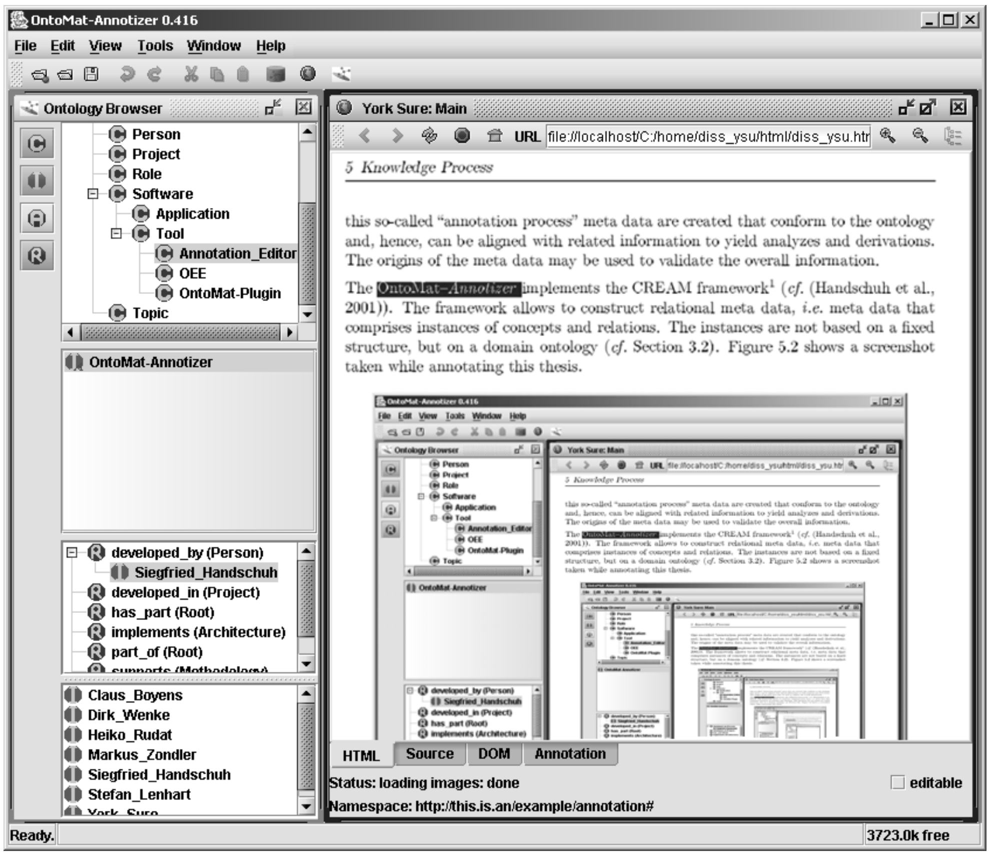
Knowledge acquisition which is a challenging area of research in artificial intelligence with its roots in early work to develop expert systems, has matured and plays a significant role for realizing the Semantic Web (*cf.*, *e.g.*, (Gómez-Pérez & Benjamins, 2002)). However, as *e.g.* shown by the broad spectrum of topics at the first and second "International Conference on Knowledge Capture" (K-CAP 2001/2003, *cf.* (Gil et al., 2001; 2003)), many other research areas contribute methods and techniques for the capturing of knowledge:

> *"Although there has been considerable work in the area of knowledge capture, activities have been distributed across several distinct research communities. In machine learning, learning apprentices acquire knowledge by nonintrusively watching a user perform a task. In the human-computer interaction community, programming-by-demonstration systems learn to perform a task by watching a user demonstrate how to accomplish it. In knowledge engineering, modeling techniques and design principles have been proposed for knowledge-based systems, often exploiting commonly occurring domain-independent inference structures and reusable domain-specific ontologies.* [...] *In natural language processing, tools can process text and create representations of its knowledge content."*
> Introduction to (Gil et al., 2001)

We now focus on the capturing of knowledge by creating meta data according to ontologies (*cf.* Section 2.4), thus by exploring methods and technologies coming from knowledge acquisition and knowledge engineering which have more recently been matured into the Semantic Web research area.

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 so-called "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.

The OntoMat–*Annotizer* implements the CREAM framework[1] (*cf.* (Handschuh et al., 2001; 2002)). The framework allows to construct relational meta data, *i.e.* meta data that comprises instances of concepts and relations. The instances are not based on a fixed structure, but on a domain ontology (*cf.* Section 3.2). Figure 5.2 shows a screenshot taken while annotating this thesis.



Status: loading images: done
Namespace: http://this.is.an/example/annotation#

5 Knowledge Process

this so-called "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.

The OntoMat–*Annotizer* implements the CREAM framework[1] (*cf.* (Handschuh et al., 2001)). The framework allows to construct relational meta data, *i.e.* meta data that comprises instances of concepts and relations. The instances are not based on a fixed structure, but on a domain ontology (*cf.* Section 3.2). Figure 5.2 shows a screenshot taken while annotating this thesis.

Figure 5.2: Annotating documents with the OntoMat–*Annotizer*

E.g., as shown in the figure, the current page is annotated by marking "OntoMat–*Annotizer*" as an instance of the concept Annotation Editor of the example ontology. Furthermore we might add instances of relations, *e.g.* the instance `OntoMat-Annotizer`

[1]CREAM stands for "Creating RElational, Annotation-based Meta data".

is related to the instance `Siegfried Handschuh` (an instance of the concept Person) by the relation DEVELOPED BY. The following sample illustrates the generated instances (created in RDF, *cf.* (Lassila & Swick, 1999)).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
 xmlns="http://this.is.an/example\#">
 <Annotation_Editor
  rdf:about="http://this.is.an/example/annotation#OntoMat-Annotizer">
   <developed_by
    rdf:resource="http://this.is.an/example#Siegfried_Handschuh"
   />
 </Annotation_Editor>
</rdf:RDF>
```

Such annotations can be further used by applications to provide better retrieval results as shown in the next Section 5.4. Another approach deals with the reverse engineering of databases (*cf.* (Stojanovic et al., 2002c)) to capture their contents. Last, but not least, there already exists an approach to handel the maintenance of such annotations, for further details we refer to (Stojanovic et al., 2002b).

5.4 Knowledge Retrieval & Access

Large parts of knowledge retrieval and access from an ontology based application are performed through conventional GUI, exploiting means like information retrieval (*cf.* (Moench, 2003) for a combination of information retrieval and ontologies).

In addition, one may use the ontology to derive further views. In particular, ontologies are often exploited for navigation purposes. Thus, knowledge workers may explore what is *e.g.* 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 (such as *e.g.* Ontobroker, *cf.* Section 6.6), *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.

The example given in Figure 5.3 is taken from the OntoWeb Semantic Portal (*cf.* Section 12.2). It presents the advanced query facilities provided by the underlying DOGMA server (*cf.* (Meersman, 1999; Jarrar & Meersman, 2002; Spyns et al., 2002a)). The following functionalities showed in the figure represent advanced functionalities that ontologies enable in such portals. The given numbers correspond to the numbers given in the figure.

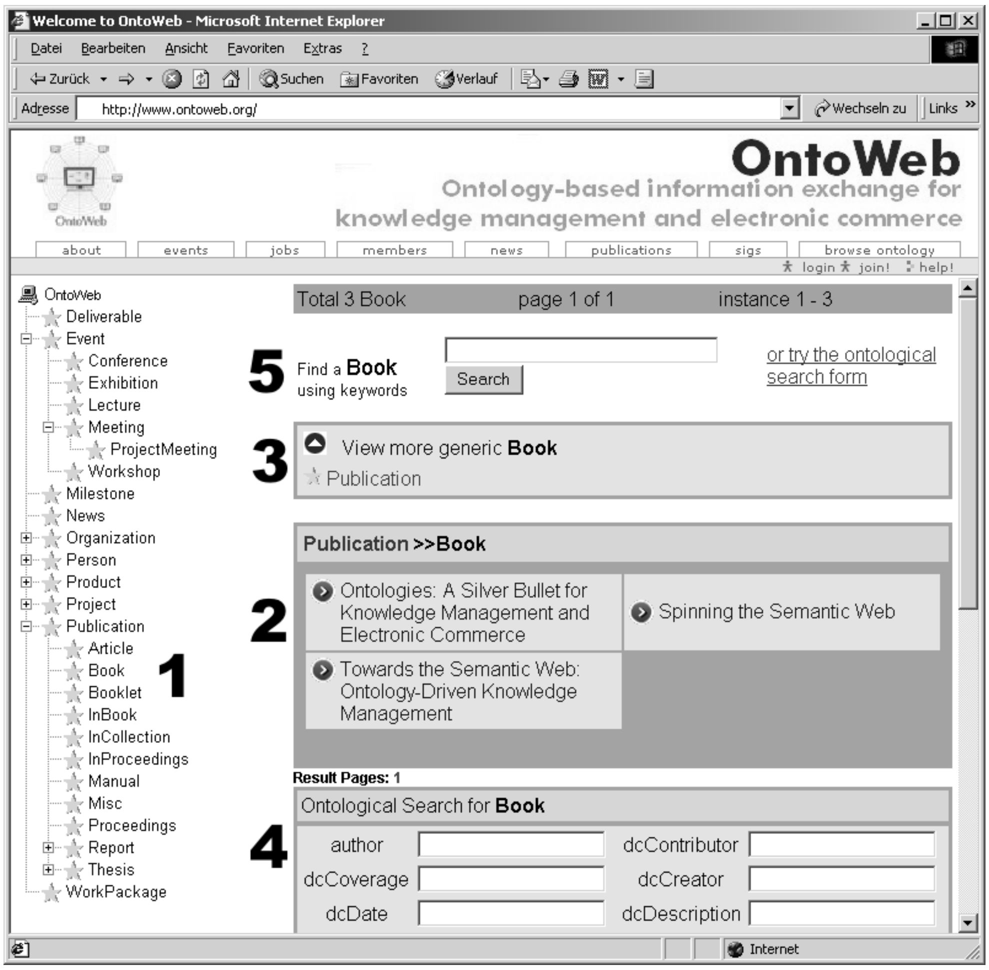

Welcome to OntoWeb - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Verlauf
Adresse http://www.ontoweb.org/
Wechseln zu Links
OntoWeb
Ontology-based information exchange for knowledge management and electronic commerce
about events jobs members news publications sigs browse ontology
login join! help!
OntoWeb
Deliverable
Event
Conference
Exhibition
Lecture
Meeting
ProjectMeeting
Workshop
Milestone
News
Organization
Person
Product
Project
Publication
Article
Book
Booklet
InBook
InCollection
InProceedings
Manual
Misc
Proceedings
Report
Thesis
WorkPackage
1
2
3
4
5
Total 3 Book page 1 of 1 instance 1 - 3
Find a Book using keywords
Search
or try the ontological search form
View more generic Book
Publication
Publication >>Book
Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce
Spinning the Semantic Web
Towards the Semantic Web: Ontology-Driven Knowledge Management
Result Pages: 1
Ontological Search for Book
author
dcContributor
dcCoverage
dcCreator
dcDate
dcDescription
Internet

Figure 5.3: Knowledge Retrieval with DOGMA in the OntoWeb Portal

1. The taxonomic structure is used for navigation in an explorer tree-like manner. Thus, users might easily browse through the ontology to explore the domain of this portal. Inherently this provides a query to the knowledge base, as shown in the next item.

2. In addition, for each concept, *e.g.* Book, the known instances are displayed when selecting it. The list directly links to further information of the instances and

possibly to downloadable resources themselves. Thus, one might also browse instances, often called the knowledge base.

3. The inherently posed query (in this case for all instances of the concept Book) can be generalized by selecting more general concepts, or

4. refined by restricting values for relations and attributes of a concept. *E.g.* one might look for instances of the concept Book that have a certain author, *i.e.* they have a certain value for the attribute AUTHOR.

5. Last, but not least, one might combine the ontology based search with a typical keyword based search.

5.5 Knowledge Use

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.

An evolving research area tackles this issue in the light of the Semantic Web, *viz.* Semantic Web Mining (*cf.* (Berendt et al., 2002b; Stumme et al., 2001; Berendt et al., 2002a)). Semantic Web Mining combines the methods of Web Mining and Semantic Web. Web Mining applies data mining techniques on the web. Three areas can be distinguished: (i) web usage mining analyzes the user behavior, (ii) web structure mining explores the hyperlink structure, and (iii) web content mining exploits the contents of the documents in the web.

The idea of Semantic Web Mining is to improve, on the one hand, the results of Web Mining by exploiting the new semantic structures in the Web; and to make use of Web Mining, on the other hand, for building up the Semantic Web.

Thus, Semantic Web Mining not only feeds again into the Knowledge Process, but also helps to make the Semantic Web become reality.

Part III

Tool Support

"One only needs two tools in life:
WD-40 to make things go,
and duct tape to make them stop."
— G. Weilacher

6 Requirements for OEEs

This part is about **tool support** for ontology engineering. In particular, we introduce plugins that realize specialized tool support for ontology engineering by extending OntoEdit, an ontology engineering environment (OEE).

Overview

We start in **this chapter** by motivating and enumerating requirements for OEEs in Section 6.1. We identify five main requirements (R1–R5). Section 6.2 illustrates the core functionalities of the OEE OntoEdit from a users perspective, including the flexible way of handling evolving ontology languages (requirement R1). The next Section 6.3 is more technically oriented and introduces the underlying OntoMat framework that guarantees a flexible extensibility (requirement R2) of OntoEdit. In Section 6.4 we give an overview of such extensions of OntoEdit, so-called "methodology plugins", that are designed to support steps of the On-To-Knowledge Methodology (requirement R3) (*cf.* Part II). We include an overview of "external plugins", *i.e.* plugins that are developed outside the scope of this work, but extend OntoEdit similar to the methodology plugins. The plugins can be divided into two sections according to their support for collaboration (R4) and inferencing (R5). The following Sections 6.5 and 6.6 introduce dimensions of collaboration and relevant inferencing issues.

The following Chapters 7, 8, 9 and 10 are dedicated to show in detail the support for steps of the Knowledge Meta Process.

6.1 Motivation of Requirements (R1 – R5)

In this part we focus on tool support for ontology engineering, *i.e.* the structured and well-defined way of developing ontologies. Ontology development is a major part of the On-To-Knowledge Methodology presented in Part II. Figure 6.1 shows the steps of the Knowledge Meta Process (*cf.* Figure 4.2) that cover ontology development. Tools that support the engineering of ontologies are often referred to as "Ontology Editors" or "Ontology Engineering Environments" (OEE). We begin this chapter by motivating and identifying typical requirements for OEEs.

During the development and application of the On-To-Knowledge Methodology in the case studies (*cf.* Part IV) we experienced practical ontology engineering. The following five requirements were identified as the major requirements for OEEs in practical settings such as described in the case studies.

Obviously, requirements known from software engineering like *e.g.* "ease of use" or "scalability" are relevant for any software. Given the focus of this thesis, we concentrate on requirements that are specific for OEEs.

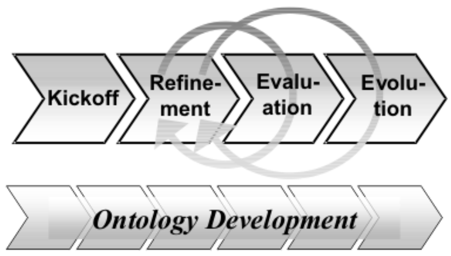


Figure 6.1: Ontology development steps of the On-To-Knowledge Methodology

Ontology development typically involves collaboration of ontology engineers and domain experts. In joint efforts they develop ontology based applications step by step in iterative cycles. A primary issue for OEEs is to support such **collaborations**, *e.g.* by facilitating distributed collaboration and guiding through the process of ontology creation. Quite naturally, OEEs should therefore also reflect steps of underlying **methodologies**.

There already exist numerous ontology languages that evolved over the last decades. Besides a growing number of languages counted by names also the version numbers of languages are growing, some of them quite fast. An example showing the current speed of evolvement is OIL, the "Ontology Inference Layer" (*cf.* (Fensel et al., 2001) and also Section 11.2.2 for a brief description). The development was initiated in the On-To-Knowledge project (*cf.* Chapter 11) in year 2000. Meanwhile, OIL has been adopted by a joined EU/US initiative that developed a language called DAML+OIL (*cf.* (Horrocks et al., 2001)), thereby combining the European OIL with the equivalent American DAML ("Darpa Agent Markup Language"). In November 2001, the W3C started a Web Ontology Working Group for defining a language. This group is chartered to take DAML+OIL as its starting point and is developing now a language called OWL, the "Ontology Web Language". There exist specialized OEEs for some languages, *e.g.* OILed (Bechhofer et al., 2001) was initially designed for OIL. More advanced OEEs like WebODE (Arpírez et al., 2001), Protégé (Noy et al., 2000) and OntoEdit, as we will show in the next section, abstract their GUIs from specific languages and provide flexible import and export facilities for a number of existing **ontology languages**.

Modularization is a typical principle in software engineering to meet changing requirements for tools. A flexible **extensibility** of such environments should therefore include import and export modules to keep pace with evolving language standards. Additionally, different usage scenarios typically require new or modified functionalities (we will show some different scenarios in the Chapters 11 and 12). Thus, extensibility should also cover customization of tools to specific scenarios.

Numerous ontology languages are based on logics and include axioms that allow for sophisticated reasoning capabilities. There exist different kinds of inference engines for different language paradigms like description logic and Frame Logic. The most well-known inference engines include *e.g.* FaCT (Horrocks, 1998) for description logic dialects and Ontobroker (Decker et al., 1999) for frame logic, *viz.* F-Logic. Firstly, OEEs might rely on such inference engines to ensure the quality of ontologies, *e.g.* to test the consistency of a concept hierarchy. Secondly, OEEs might use reasoning capabilities to develop ontologies, if they want to be able to cover all aspects of such languages, *viz.*, *e.g.*, for developing and testing axioms. **Inferencing** support is a valuable add-on for ontology engineering environments.

In the following sections and chapters we will illustrate how OntoEdit meets the requirements summed up in Table 6.1.

Table 6.1: Summary of the requirements for Ontology Engineering Environments

R1	Ontology Languages
R2	Extensibility
R3	Methodology
R4	Collaboration
R5	Inferencing

R4: Collaboration and **R5: Inferencing** can be seen as essential parts of the support for **R3: Methodology**. As a core contribution of this thesis we present in the following chapters of this part specialized tool support for the steps of the Knowledge Meta Process and explain which requirements they fulfill.

Next in this chapter, we will (i) introduce in Section 6.2 the core ideas of OntoEdit as an ontology engineering environment, *e.g.* by showing its ability to import and export current and future evolving ontology languages and standards (requirement **"R1: Ontology Languages"**), (ii) show in Section 6.3 the extensibility of OntoEdit through the underlying OntoMat plugin framework (requirement **"R2: Extensibility"**), and (iii) introduce in Section 6.4 the "methodology plugins" that are specifically designed to support steps of the methodology (requirement **"R3: Methodology"**) and distinguish them according to their support for **"R4: Collaboration"** or **"R5: Inferencing"**. Last, but not least, we will elaborate more on the foundations for collaboration and inferencing by introducing (iv) the dimensions of collaboration in Section 6.5 and (v) theoretical and practical issues of inferencing in Section 6.6.

6.2 Ontologies in OntoEdit (R1)

In this section we introduce the core functionalities of OntoEdit. Thereby, we also describe OntoEdit's facility to flexibly support different ontology languages, *i.e.* the support for requirement **"R1: Ontology Languages"**.

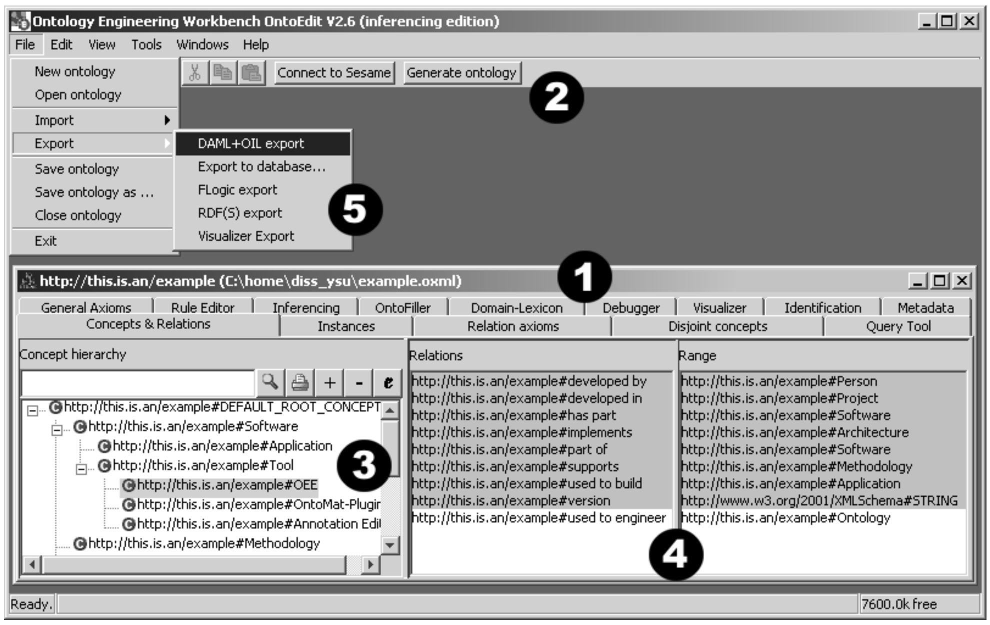
OntoEdit is an ontology engineering environment which allows for inspecting, browsing, codifying and modifying ontologies. Modeling ontologies using OntoEdit means modeling at a conceptual level, *viz.* (i) as much as possible independent of a concrete representation language, (ii) using graphical user interfaces (GUI) to represent views on conceptual structures, *i.e.* concepts ordered in a concept hierarchy, relations with domain and range, instances and axioms, rather than codifying conceptual structures in ASCII.

We now explain the main GUI of OntoEdit[1]. To put the description into a wider perspective, *viz.* the *Karlsruhe Perspective on Ontologies* (*cf.* (Bozsak et al., 2002)), we give reference links to this ontology definition. We begin with the elements for (i) modeling concepts and relations and (ii) importing and exporting ontologies in different ontology languages as shown in Figure 6.2.

The following numbers refer to the corresponding numbers shown in this figure.

1. This window represents an ontology opened in OntoEdit. Each ontology is identified uniquely by a URI (Uniform Resource Identifier, defined in (Berners-Lee, 1993) as "generic set of all names/addresses that are short strings that refer to resources"). In our example we use the URI `http://this.is.an/example`. Each tab shown provides specialized functionalities that are encapsuled in modules called "plugins" (*cf.* next section for the underlying plugin framework called "OntoMat"), *e.g.* here the tab 'Concepts & Relations' is opened.

[1]Further information about OntoEdits historical roots and a complete description of the main GUI can be found in Appendix A.



Relations	Range
http://this.is.an/example#developed by	http://this.is.an/example#Person
http://this.is.an/example#developed in	http://this.is.an/example#Project
http://this.is.an/example#has part	http://this.is.an/example#Software
http://this.is.an/example#implements	http://this.is.an/example#Architecture
http://this.is.an/example#part of	http://this.is.an/example#Software
http://this.is.an/example#supports	http://this.is.an/example#Methodology
http://this.is.an/example#used to build	http://this.is.an/example#Application
http://this.is.an/example#version	http://www.w3.org/2001/XMLSchema#STRING
http://this.is.an/example#used to engineer	http://this.is.an/example#Ontology

Figure 6.2: Basic functionalities of OntoEdit

2. Further functionalities (also provided by specialized plugins) can be accessed *e.g.* through buttons in the main OntoEdit window. Shown here is *e.g.* the "Generate Ontology" button that opens the "OntoGenerator" (*cf.* Section 9.1).

3. The "Concepts & Relations" tab allows for creating, editing and deleting concepts and relations. Creating concepts and relations includes (i) creating internally unique identifiers that are not visible to users, (ii) creating signs in different languages, also called "external representations", (*cf.* the handling of multiple languages in Appendix A) that are displayed in the GUI and (iii) creating and maintaining references between unique identifiers and external representations. Thus, OntoEdit automatically creates an *ontology with lexicon* (*cf.* (Bozsak et al., 2002)). Concepts are ordered in an inheritance hierarchy, a so-called concept hierarchy. By default, all names of created concepts get attached the URI of the ontology as a namespace to ensure their unique identity. In Figure 6.2, concept OEE is selected (indicated by the highlighting). OEE "is a" Tool, *i.e.* OEE is a direct subconcept of Tool and inherits all its relations.

4. All relations that have OEE as their domain are shown *without highlighting* in this figure. All relations that OEE inherits from its superconcepts are shown *with highlighting*. *E.g.*, (i) the relation USED TO ENGINEER with the domain Ontology

has the range OEE, (ii) the relation DEVELOPED BY has the range Software and Software is a superconcept of OEE. By default, all created relations get attached as a namespace the URI of the ontology to ensure their unique identity. However, concerning a relation hierarchy similar to the concept hierarchy (*cf.* (Bozsak et al., 2002)) is not yet implemented in OntoEdit, *i.e.* relations currently form a flat structure.

5. The conceptual model of an ontology is internally stored using a powerful ontology model (it conforms mainly with the definitions in (Bozsak et al., 2002), only excluding the relation hierarchy). To meet requirement **"R1: Ontology Languages"** , this ontology model can be mapped onto different state-of-the-art representation languages. The mapping is implemented as import and export filters, *e.g.* as highlighted in Figure 6.2, for DAML+OIL[2] (Horrocks et al., 2001). Further import/export filters exist *e.g.* for F-Logic (Kifer et al., 1995) and RDF(S) (Brickley & Guha, 2002).

In Figure 6.3 the "Instances" tab is shown that allows for modelling of instances in OntoEdit.

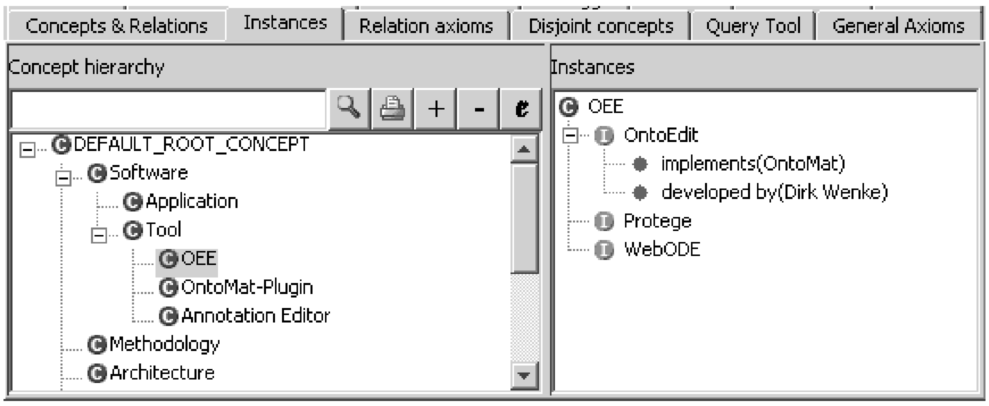


Figure 6.3: Modeling instances with OntoEdit

The tab consists of two main elements. On the left side the concept hierarchy is shown (similar to the "Concepts & Relations" tab in Figure 6.2). One may select a concept to show its *concept instantiations* (*cf.* (Bozsak et al., 2002)). Here the concept OEE is selected. On the left side the concept instantiations are shown. Equivalent to the handling of concepts and relations, OntoEdit creates automatically internal identifiers for instances as well as it creates and manages the references to signs shown in the

[2]A detailed description of DAML+OIL features supported by OntoEdit can be found in Appendix C.

GUI. *E.g.* shown in the figure are the signs for the instances `OntoEdit`, `Protégé` and `WebODE`. Attached to each instance are its belonging *relation instantiations* (*cf.* (Bozsak et al., 2002)). *E.g.* shown in the figure is the relation instantiation for the relation DEVELOPED BY with an instance of Person, *viz.* `Dirk Wenke`, as value for the range.

6.3 Extensibility through the OntoMat Framework (R2)

In this section we illustrate how OntoEdit meets requirement **"R2: Extensibility"** (*cf.* Table 6.1). We start with an introduction to the plugin framework OntoMat, show the advantages of the framework, describe the basic functionalities provided in OntoEdit by the framework and, last but not least, illustrate different types of plugins in OntoEdit.

Introduction to the OntoMat Framework

Building up from standardized units to complex things and products is a well-known construction principle from engineering disciplines. With the development of software systems this concept gains ever more significance.

OntoEdit is based on a flexible plugin framework called OntoMat (Handschuh, 2001). The OntoMat framework offers the possibility to implement functions by extensions. Such extensions are usually called snap-ins, add-ons or plugins. The term "plugin" in the context of the framework means a software component, which implements the appropriate Java interface. The framework consists of a core for the plugin management and a general plugin which represents the core functionality. All additional functionality is totally enclosed in additional plugins.

All plugins are dynamically plug-able during runtime. The plugin mechanism notifies each installed component, when a new component is registered. Through the service mechanism each component can discover and utilize the services offered by another component. A service represented by a component is typically a reference to an interface. This provides among other things a de-coupling of the service from the implementation and allows therefore alternative implementations.

Advantages of the OntoMat Framework

The main advantages of this architectural idea are:

- **Extensibility:** The system can be extended by plugins. Each new functionality can be realized as independent plugin. The interoperability between the plugins is realized over services. Each plugin can offer new services.

- **Flexibility:** Each unnecessary plugin can be removed and each necessary plugin can be loaded at run-time. Therefore the system can be configured in such a way that it has only the needed functionality to adapt OntoEdit to different usage scenarios.
- **De-coupling:** The service mechanism provides a de-coupling of the service from the implementation and allows therefore alternative implementations.
- **Modularity:** Plugins may access over the service mechanism all other plugins. The interface of the service reveals as little as possible about the inner working of an service. This isolates the client of the services from requiring intimate knowledge of the design of the service, and from the effects of changing those decisions. This information hiding is a primary criteria for system modularization.

Basic Functionalities of OntoEdit

The basic functionalities of OntoEdit are realized as so-called "**core plugins**" that provide different functionalities for the management and hosting of additional plugins and services.

- OntoEdit realizes a "Multiple Document Interface" (MDI), with some standard menu entries.
- Plugin administration consists of two main functionalities:
  - A plugin **Management Console**. The plugins are registered with this console. Users must indicate the complete class name of the plugin for that. The console loads and unloads the registered plugins. Furthermore it indicates the copyright messages and possible error messages of plugins. Each plugin is examined with registering. It is discovered whether it offers a service or it would like using a service.
  - The **Option Manager** shows a list of the installed plugins (*cf.*, *e.g.*, Appendix B). If a plugin from the list is selected the associate option panel will be visible. Each plugin that has options can offer such a panel.
- All additional plugins implement certain Java interfaces in order to be plug-able into the framework and to offer additional services.
- Besides of the service mechanism OntoEdit has further means of internal communication. OntoEdit produces internal standard events, *e.g.* clipboard events or file events. These events are dispatched and each plugin can utilize these events.

Each plugin can provide menu entries within the menu bar and an icon within the toolbar. The menu entries and the icon are connected with actions of the plugin. Apart from the own menu actions a plugin can react to an OntoEdit standard event, if it has registered itself for it. Further, it can share an event model with another plugin based on the service mechanism.

A plugin can be in the role of a **service-provider** or a **service-consumer**, according to the Java interface it implements. A plugin that implements the service-consumer interface registers it and intends to be notified of new services. On the other hand a plugin that implements the service-provider interface register a new service with the framework. The framework notifies all currently registered service-user that this new service-provider plugin has been added. After being notified of the newly available service, the listening plugin requests an instance of the service from the framework. The framework tells the service-provider to deliver the service to the requesting plugin.

Each plugin has access to the MDI Desktop of OntoEdit. It can independently place windows there. Also a data exchange is possible between these plugin windows by drag and drop if these windows are appropriately implemented.

Types of Plugins in OntoEdit

Plugins themselves are not restricted in their potential functionalities, but in OntoEdit they typically can be categorized into four main types of functionalities like shown in Figure 6.4.

- **Core plugins** are needed to build up internal structures like the datamodel or other basic functionalities (see previous section).
- **GUI plugins** extend the core plugins by adding GUI elements that represent specific functionalities, *e.g.* the methodology plugins (see Section 6.4) support specific steps of the Knowledge Meta Process (see Chapter 4).
- **Connector plugins** realize connections to databases or inference engines like Ontobroker (see Section 6.6).
- **Parser plugins** realize the import and export filters for numerous state-of-the-art representation languages mentioned in the previous Section 6.2.

Appendix B illustrates the implementation of OntoEdit on top of OntoMat more detailed, it *e.g.* includes descriptions of the common OntoMat interfaces and of OntoEdit's core plugins.

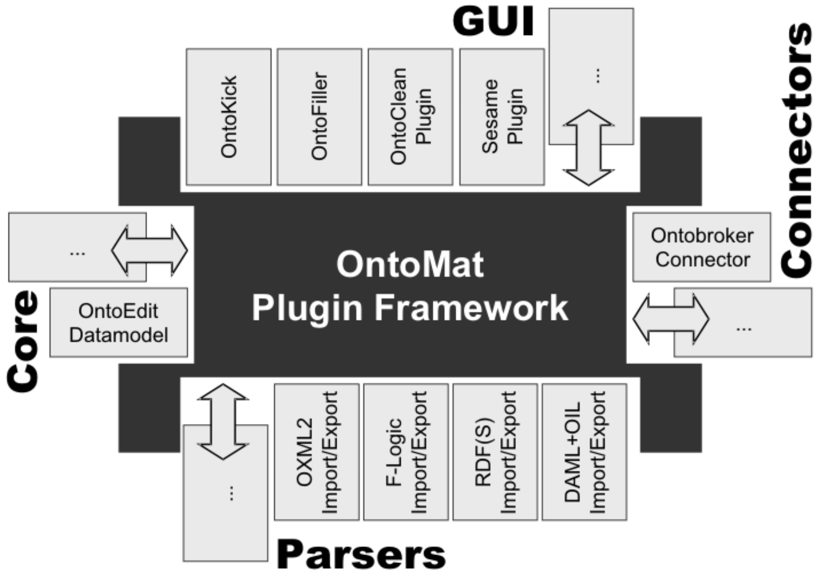


Figure 6.4: Plugin architecture of OntoEdit

6.4 Methodology Support (R3)

In this section we give an overview on how OntoEdit meets requirement **"R3: Methodology"** (*cf.* Table 6.1) by introducing the so-called "methodology plugins". Figure 6.5 presents the "methodology plugins". These plugins were initiated and implemented to support the steps of the On-To-Knowledge Methodology. Typically each plugin is designed to support a specific step.

The plugins can be divided into two groups according to their support for **"R4: Collaboration"** or **"R5: Inferencing"**.

There exist already numerous further plugins for OntoEdit (*cf.* (Ontoprise, 2002b)). They are developed outside the scope of this work, but extend OntoEdit similar to the methodology plugins. They tackle common problems known in the area of ontology engineering. These plugins, in this work referred to as "external plugins", can also be categorized by their support for steps of the On-To-Knowledge Methodology and their support for **R4: Collaboration** or **R5: Inferencing**. Figure 6.6 gives an overview of existing external plugins for OntoEdit.

At the end of each of the following Chapters 7, 8, 9 and 10 we will briefly describe the functionality of the "external plugins". The intention is to present a rather complete picture how OntoEdit currently meets **"R3: Methodology"**, **"R4: Collaboration"**

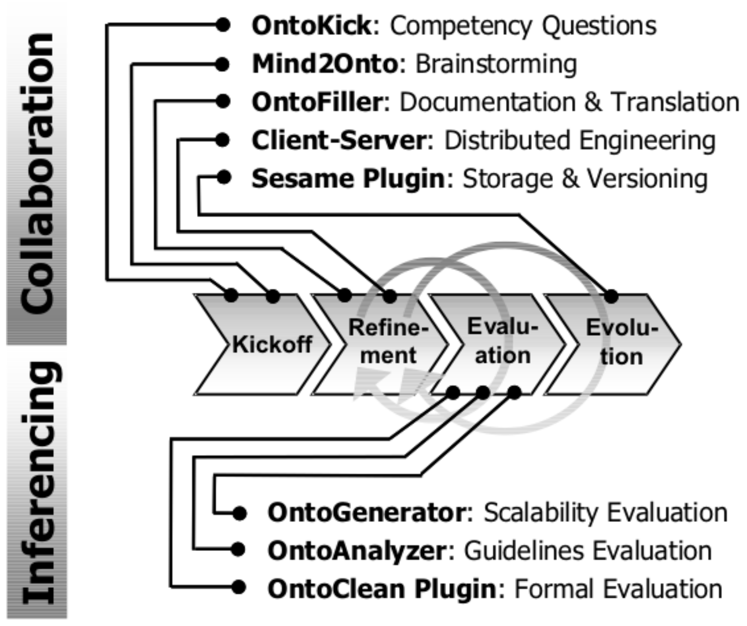

Collaboration
Inferencing
OntoKick: Competency Questions
Mind2Onto: Brainstorming
OntoFiller: Documentation & Translation
Client-Server: Distributed Engineering
Sesame Plugin: Storage & Versioning
Kickoff
Refine-ment
Evalu-ation
Evolu-tion
OntoGenerator: Scalability Evaluation
OntoAnalyzer: Guidelines Evaluation
OntoClean Plugin: Formal Evaluation

Figure 6.5: Overview of the "methodology plugins"

and **"R5: Inferencing"**(*cf.* Table 6.1).

6.5 Dimensions of Collaboration (R4)

Computer-Supported Cooperative Work (CSCW) deals with systems that support the collaboration of people, as mentioned in the following quotation.

> *"CSCW looks at how groups work and seeks to discover how technology (especially computers) can help them work."*
> (Ellis et al., 1991)

(Nichols & Twidale, 1999) divide the dimensions according to the terms of the time and space in which a collaborative activity occurs. Collaboration can be between people

- in the *same place* or *different places*,
- at *the same time* or *separated in time.*

Also, as further mentioned in (Nichols & Twidale, 1999), collaboration can be (in libraries) (i) among staff members, (ii) among users and staff members, and (iii) among

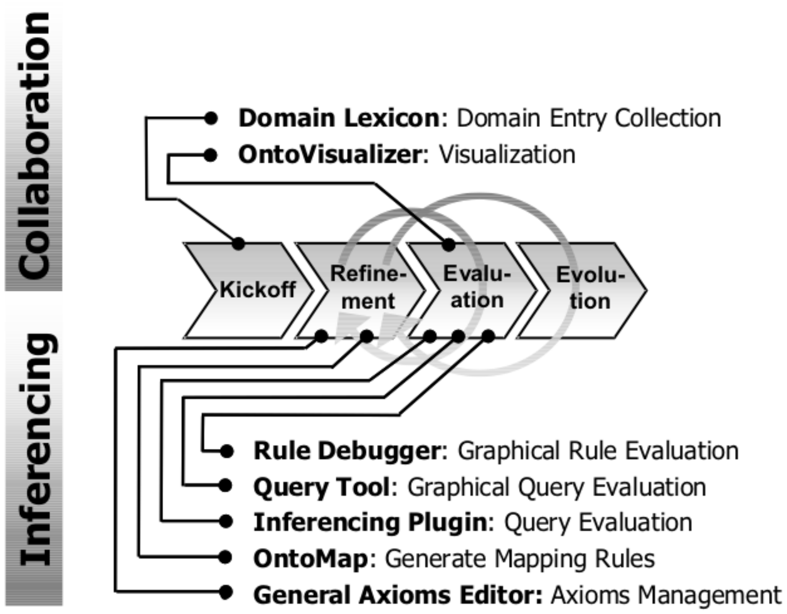

Collaboration
Inferencing
Domain Lexicon: Domain Entry Collection
OntoVisualizer: Visualization
Kickoff
Refine-ment
Evalu-ation
Evolu-tion
Rule Debugger: Graphical Rule Evaluation
Query Tool: Graphical Query Evaluation
Inferencing Plugin: Query Evaluation
OntoMap: Generate Mapping Rules
General Axioms Editor: Axioms Management

Figure 6.6: Overview of "external plugins"

users. Similarly, in ontology engineering we distinguish between the collaboration among

- *domain experts*,
- *ontology engineers* & *domain experts*, and
- *ontology engineers*.

OntoEdit was not designed from the very beginning to be a cooperative tool. Figure 6.7 gives an overview of additional "collaboration plugins" that were developed to support the collaboration task. The figure includes all "methodology plugins" and "external plugins" that support collaboration (*cf.* Section 6.4).

Characterization of Plugins

We will introduce each of the plugins in the subsequent chapters of this part. At the beginning of each section the plugins will be characterized *e.g.* according to their support for collaboration or inferencing. For each plugin we will depict

- the **steps** of the Knowledge Meta Process that are supported,

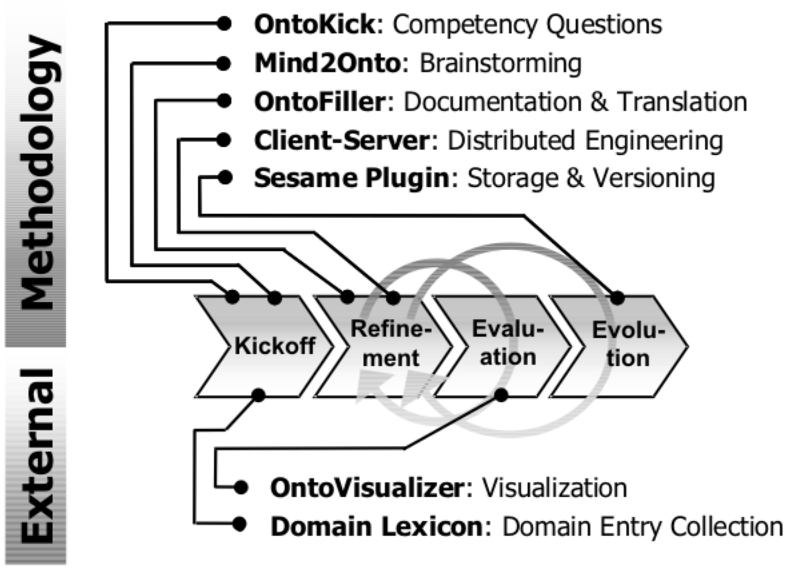

Methodology
External
OntoKick: Competency Questions
Mind2Onto: Brainstorming
OntoFiller: Documentation & Translation
Client-Server: Distributed Engineering
Sesame Plugin: Storage & Versioning
Kickoff
Refine-ment
Evalu-ation
Evolu-tion
OntoVisualizer: Visualization
Domain Lexicon: Domain Entry Collection

Figure 6.7: Overview of the "collaboration plugins"

- the different **groups** that are supported, and
- the **tasks** that are supported and **benefits** that are achieved.

For all plugins that support collaboration we add

- the supported **time and space dimensions**.

Existing external plugins are briefly sketched at the end of each chapter.

6.6 Theoretical & Practical Inferencing Issues (R5)

In this section we introduce theoretical and practical issues of inferencing including the logical framework underlying OntoEdit and its attached inference engine Ontobroker. We start with an introduction to relevant inferencing issues, explain the underlying logical framework of OntoEdit and Ontobroker and, last but not least, enumerate some functionalities of Ontobroker.

Introduction

While OntoEdit is primarily used to engineer ontologies, Ontobroker is able to process rules, also called axioms (we use both concepts synonymously), and draw inferences, *i.e.* to operationalize ontologies which makes Ontobroker suitable as a backend for ontology based applications. However, Ontobroker is tightly integrated into OntoEdit (via a "connector plugin", *cf.* Section 6.3) and therefore is also suitable to support ontology engineering tasks.

Figure 6.8 shows an overview of the "inferencing plugins". The plugins will be described in the subsequently following sections of this chapter. Two major use cases are relevant for the usage of inferencing for ontology engineering:

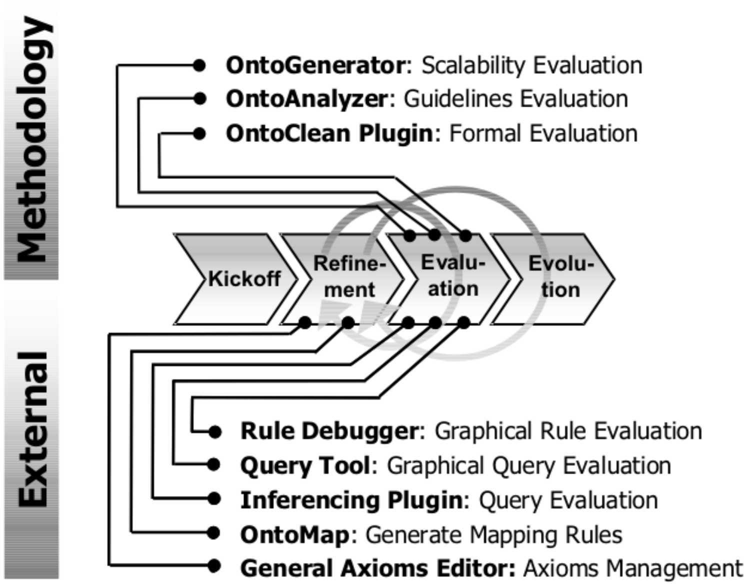

Methodology
OntoGenerator: Scalability Evaluation
OntoAnalyzer: Guidelines Evaluation
OntoClean Plugin: Formal Evaluation
Kickoff
Refine-ment
Evalu-ation
Evolu-tion
External
Rule Debugger: Graphical Rule Evaluation
Query Tool: Graphical Query Evaluation
Inferencing Plugin: Query Evaluation
OntoMap: Generate Mapping Rules
General Axioms Editor: Axioms Management

Figure 6.8: Overview of the "inferencing plugins"

1. **Application** of pre-defined axioms to an ontology, *e.g.* for evaluation purposes.
2. **Creation** of axioms as part of the ontology.

The "methodology plugins" support the first use case (as shown in the following Sections 9.1, 9.2 and 9.3). The "external plugins" support (mainly) the second use case (as briefly described in Sections 8.3 and 9.4). In a nutshell the "external plugins" support the following (sub-) use cases:

- Text-based and (more advanced) graphically oriented creation of axioms.

- Text-based and (more advanced) graphically oriented (creation and) evaluation of queries that can be *e.g.* used for applications, and
- evaluation of axioms and queries by switching them on/off during the query processing.
- Graphically oriented debugging of axioms.
- Creation of mappings between ontologies, the mappings are expressed as axioms and can be processed with Ontobroker *e.g.* during the runtime of an application.

More examples for the usage of inferencing during ontology engineering can be found in (Sure et al., 2002b).

Logical Framework

In order to provide a clearly defined semantics to the knowledge model of OntoEdit, the knowledge structures of OntoEdit correspond to a well-understood logical framework, *viz.* F-Logic (*cf.* (Kifer et al., 1995), "F" stands for "Frames"). F-Logic combines deductive and object-oriented aspects:

> *"F-logic* [...] *is a deductive, object-oriented database language which combines the declarative semantics of deductive databases with the rich data modelling capabilities supported by the object oriented data model."*
> (Frohn et al., 1996)

F-Logic allows for concise definitions with object oriented-like primitives (classes, attributes, object-oriented-style relations, instances) that are reflected by the OntoEdit GUI. Furthermore, it also has Predicate Logic (PL-1) like primitives (predicates, function symbols), that are only partially reflected in the GUI but internally used within the data structures. F-Logic allows for axioms that further constrain the interpretation of the model. Axioms may either be used to describe constraints or they may define rules, *e.g.* in order to define a relation R by the composition of two other relations S and Q.

F-Logic rules have the expressive power of Horn-Logic with negation and may be transformed into Horn-Logic rules. The semantics for a set of F-Logic statements is defined by the well-founded semantics (*cf.* (van Gelder et al., 1991)). This semantics is close to First-Order semantics. In contrast to First-Order semantics not all possible models are considered but one "most obvious" model is selected as the semantics of a set of rules and facts. It is a three valued logic, *i.e.* the model consists of a set of true facts and a set of unknown facts and a set of facts known to be false.

In comparison to other logic-based representation languages for ontologies, F-Logic is quite well suited for the usage within the World Wide Web (*cf.* (Decker, 2002)). Unlike Description Logics (DL), F-Logic does not provide means for subsumption (Horrocks, 1998), but (also unlike DL) it provides for efficient reasoning with instances and for the capability to express arbitrary powerful rules, *e.g.* ones that quantify over the set of classes.

The most widely published operational semantics for F-Logic is the alternating fixed point procedure. This is a forward chaining method which computes the entire model for the set of rules, *i.e.* the set of true and unknown facts. For answering a query the entire model must be computed (if possible) and the variable substitutions for the query are then derived. In contrast, the inference engine Ontobroker performs a mixture of forward and backward chaining based on the dynamic filtering algorithm (*cf.* (Kifer & Lozinskii, 1986)) to compute (the smallest possible) subset of the model for answering the query. In most cases this is much more efficient than the simple evaluation strategy. These techniques stem from the deductive data base community and are optimized to deliver all answers instead of one single answer as *e.g.* resolution does.

Within the F-Logic compiler F-Logic statements are translated to normal programs. Normal programs are Horn programs where rules may contain negated literals in their bodies. Horn Logic is Turing Complete, thus F-Logic programs are not decidable in principle. The semantics defined for these normal programs is the well-founded semantics (van Gelder, 1993). In contrast to the stratified semantics the well-founded semantics is also applicable for rules which depend on cycles containing negative rule bodies. Because F-Logic is very flexible, during the translation to normal programs such negative cycles often arise. In (van Gelder et al., 1991) the alternating fixpoint has been described as a method to operationalize such logic programs. This method has been shown to be very inefficient. Therefore the inference engine realizes dynamic filtering (Kifer & Lozinskii, 1986) which combines top-down and bottom-up inferencing. Together with an appropriate extension to compute the well-founded semantics this method has been proven to be very efficient compared to other horn based inference engines (*cf.*, *e.g.*, (Sure et al., 2002b)).

For detailed introductions to the syntax and the object model of F-Logic, in particular with respect to the implementation of F-Logic in Ontobroker, we refer to (Erdmann, 2001), (Decker, 2002) and (Ontoprise, 2002a).

Inference Engine Ontobroker

The inference engine Ontobroker (*cf.* (Decker et al., 1999)) was developed in numerous years of scientific research (*cf.*, *e.g.*, (Angele, 1993; Fensel, 1995; Erdmann, 2001;

Decker, 2002)) and is now being commercialized by the company Ontoprise[3]. It comes with several features that makes it adequate not only for inferencing purposes but rather as a backbone for an ontology editor. In particular, it provides:

- A namespace mechanism: Thus, several ontologies (or ontology parts) may be syntactically split into modules and processed by different inference engines.
- Switch-off: It is possible to switch of (possibly singleton) sets of definitions. Thus, one may test interactions and easily distinguish between modules.
- Database Connectors: Thus, one may easily map db tables into predicates via, *e.g.*, JDBC.
- User-definable Built-ins: Besides of standard built-ins like "multiply", the user may define his own ones for special purposes.
- An extensive API: Thus, one may remotely connect to the inference engine and one may also import and export several standards (*e.g.*, RDF(S)).

Characterization of Plugins

As mentioned in the previous section, we will introduce each of the plugins in the subsequent chapters of this part. At the beginning of each section the plugins will be characterized *e.g.* according to their support for collaboration or inferencing. For each plugin we will depict

- the **steps** of the Knowledge Meta Process that are supported,
- the different **groups** that are supported, and
- the **tasks and benefits**.

For all plugins that support inferencing we add

- whether they support the **creation** or **application** of axioms.

Existing external plugins are briefly sketched at the end of each chapter.

[3]Ontoprise GmbH, see `http://www.ontoprise.de/`

7 Kickoff Support

Overview

In **this chapter** we present specialized plugins that support the kickoff phase of the Knowledge Meta Process. In Section 7.1 we show how to capture competency questions and other requirements with OntoKick. The Mind2Onto framework described in Section 7.2 integrates brainstorming into the early stages of ontology engineering. Both plugins provide support for **R3: Methodology** and **R4: Collaboration**. Last, but not least, we briefly introduce further support for the kickoff phase by external plugins in Section 7.3.

Note: The ontology that is used as an illustrating example is included completely in Appendix D.

References: This chapter is mainly based on parts of (Sure et al., 2002a).

7.1 Competency Questions (and other Requirements)

As described in Section 4.4, the actual development of the ontology begins in the Kickoff phase. Similar to software engineering and as proposed by (Fernández-López 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 next step, the gathering of the semi-formal description of the ontology.

OntoKick extends the functionality of OntoEdit by support for capturing requirements specifications for an ontology based application. It focusses on two aspects: (i) the capturing of meta aspects of an ontology during the kick–off in an "electronic" ORSD that is stored along with ontologies and (ii) the capturing of competency questions during the kick–off and the extraction of concepts, relations and instances during the refinement. The OntoKick plugin is characterized in Table 7.1.

Capturing meta aspects: OntoKick allows for describing meta aspects of the ontology such as the domain and the goal of the ontology, potential design guidelines,

Table 7.1: OntoKick characterization

Methodology:	Kick–off (and partially refinement)
Task:	Capturing and storing competency questions and requirements
Benefits:	Traceability and context
Requirements:	Methodology (R3) and Collaboration (R4)
Space:	Same place, different place
Time:	Same time, separated in time
Group:	Ontology engineers, ontology engineers & domain experts

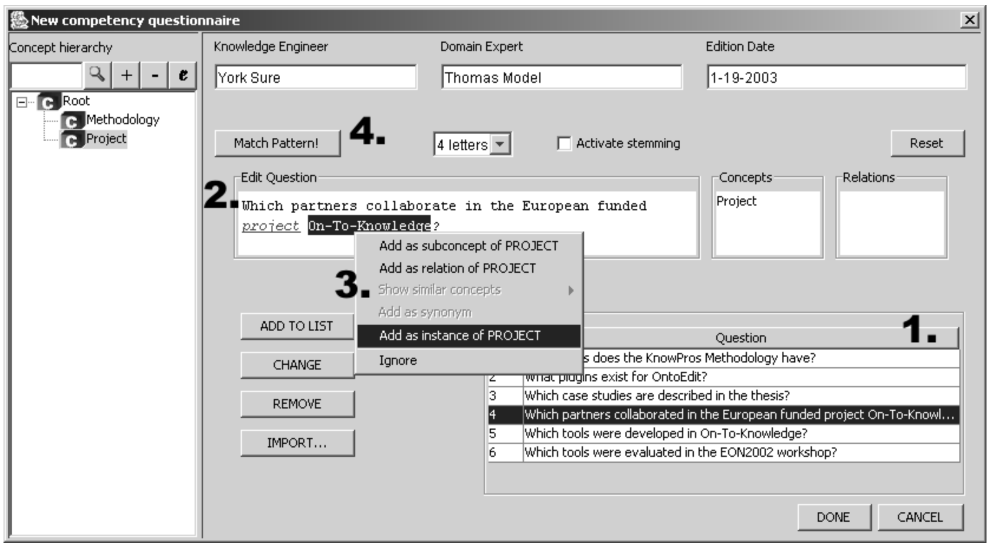
available knowledge sources (*e.g.* contact data of domain experts, pointers to known reusable ontologies *etc.*), potential users and use cases, and applications supported by the ontology. OntoKick guides a user through all relevant aspects and stores these descriptions along with the ontology definitions, thereby making it available at any time and place for ontology engineers.

Capturing competency questions: As proposed by (Uschold & King, 1995), 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. One can say that an ontology should have the "competence" to answer such questions. 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 and the context of concept definitions in later stages. Figure 7.1 shows an example of a CQ in OntoEdit.

The numbers in the figure refer to the following steps that are taken during an interview session when using OntoKick. Typically an ontology engineer and a domain expert take part in such a session.

1. During the kick–off an ontology engineer captures a list of relevant competency questions from a domain expert, *e.g.* the CQ "Which partners collaborate in the European funded project On-To-Knowledge?".



Knowledge Engineer: York Sure
Domain Expert: Thomas Model
Edition Date: 1-19-2003
4 letters

Edit Question: Which partners collaborate in the European funded project On-To-Knowledge?

Concepts: Project

Relations:

	Question
	s does the KnowPros Methodology have?
2	What plugins exist for OntoEdit?
3	Which case studies are described in the thesis?
4	Which partners collaborated in the European funded project On-To-Knowl...
5	Which tools were developed in On-To-Knowledge?
6	Which tools were evaluated in the EON2002 workshop?

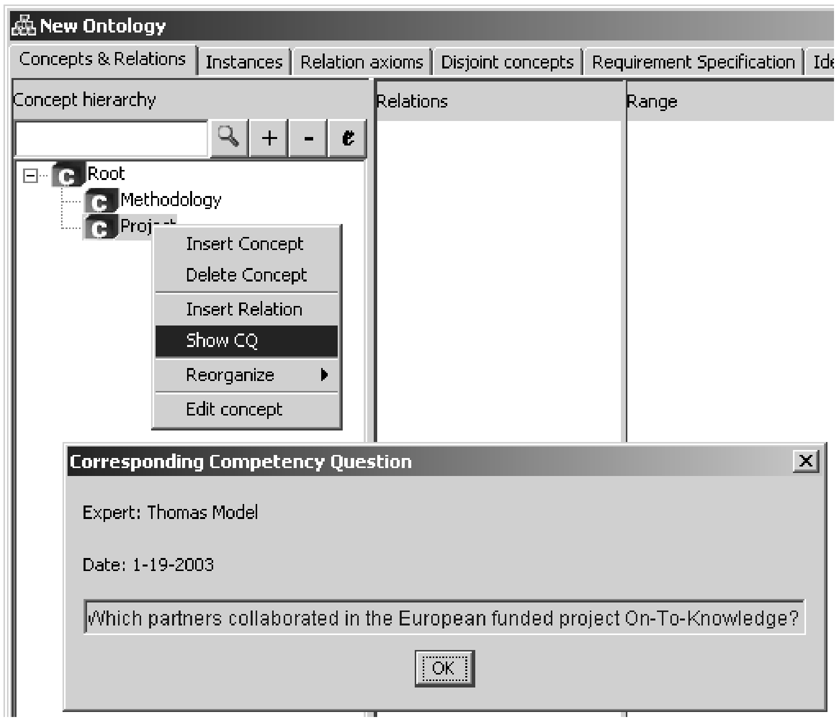
Figure 7.1: Capturing competency questions with OntoKick

2. During the refinement ontology engineers can extract concepts, relations and instances out of the competency questions. *E.g.* the concepts Methodology and Project are already modelled.

3. On-To-Knowledge can be inserted as an instance of the selected concept Project (or any other already existing concept) by marking it up and using a context menu. Also, a marked item can be inserted as a subconcept of a concept or as a relationship with having the domain of an already given concept. When using a domain lexicon, one might add the marked item as a synonym for a concept.

4. When an ontology is quite big, it might be difficult to decide whether *e.g.* a concept already has been modelled or not. A simple pattern matching facility (including a stemming facility for the English language) allows for finding syntactically similar concepts by comparing a pre-selected number of characters (or letters) of the marked up item to letters in the names of existing concepts. *E.g.* when marking up "project" and selecting "3 letters" for pattern matching, one would obviously match "project", but also, *e.g.*, "profile". Both contain *e.g.* the three letters "pro". When marking up "On-To-Knowledge" and selecting "9 letters" for pattern matching, one would *e.g.* also match "Knowledge".

During later stages of refinement, it might be helpful to trace back why concepts, relations or instances have been modelled as they are (*cf.* (Landes, 1995)), especially when multiple ontology engineers work on the same ontology. OntoKick keeps track of extracted concepts and their corresponding competency questions (in a future version this is also planned for relations and instances). Figure 7.2 shows the tracing from a concept to the corresponding CQ. This provides not only the origins – and thereby some context – of concepts, but also links to domain expert who *e.g.* can be contacted for further questions ontology engineers might have.



Expert: Thomas Model

Date: 1-19-2003

Which partners collaborated in the European funded project On-To-Knowledge?

Figure 7.2: Traceability of concepts with OntoKick

Outlook: As mentioned above, the traceability is currently only available for concepts, but is planned also for relationships and instances. Other possible add-ons include *e.g.* the linking of OntoKick with existing contact data managing applications such as Outlook to make the collaboration with relevant persons as easy as possible. A linking to search engines such as the knowledge retrieval platform Semantic Miner (*cf.* (Moench, 2003)) might help in finding people if the contact data are not sufficient.

7.2 Brainstorming

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 sub-branches. 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.* (Lau & Sure, 2002)).

The Mind2Onto framework integrates brainstorming into the kickoff phase of the Knowledge Meta Process. It is characterized in Table 7.2.

Table 7.2: Mind2Onto characterization

Methodology:	Kick–off (and partially refinement)
Task:	Brainstorming, capturing domain knowledge
Benefits:	Easy to use, intuitive, understandable
Requirements:	Methodology (R3) and Collaboration (R4)
Space:	Same place, different place
Time:	Same time, separated in time
Group:	Domain experts, ontology engineers & domain experts

In our case studies (*cf.* Sections 11.4 and 11.5) we typically performed the following three steps while using mind maps™:

1. During the kick–off the domain experts create initial mind maps™ about their domain (typically without the help of ontology engineers).
2. During the refinement the domain experts and ontology engineers jointly refine mind maps™.
3. Finally, during the refinement, the ontology engineers transform and formalize the mind maps™ into ontologies.

We rely on a widely used commercial tool[1] for the creation of mind maps™. Figure 7.3 shows an example mind map™ that contains the most relevant concepts for this work.

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

Noteworthy is that certain concepts, *e.g.* "Methodology", have attached codes, *e.g.* like the check mark which means "Meilenstein" (German for "milestone"), and/or have a separate layout like different fonts or colors. This mind map™ has already be sorted to reflect a taxonomical structure.

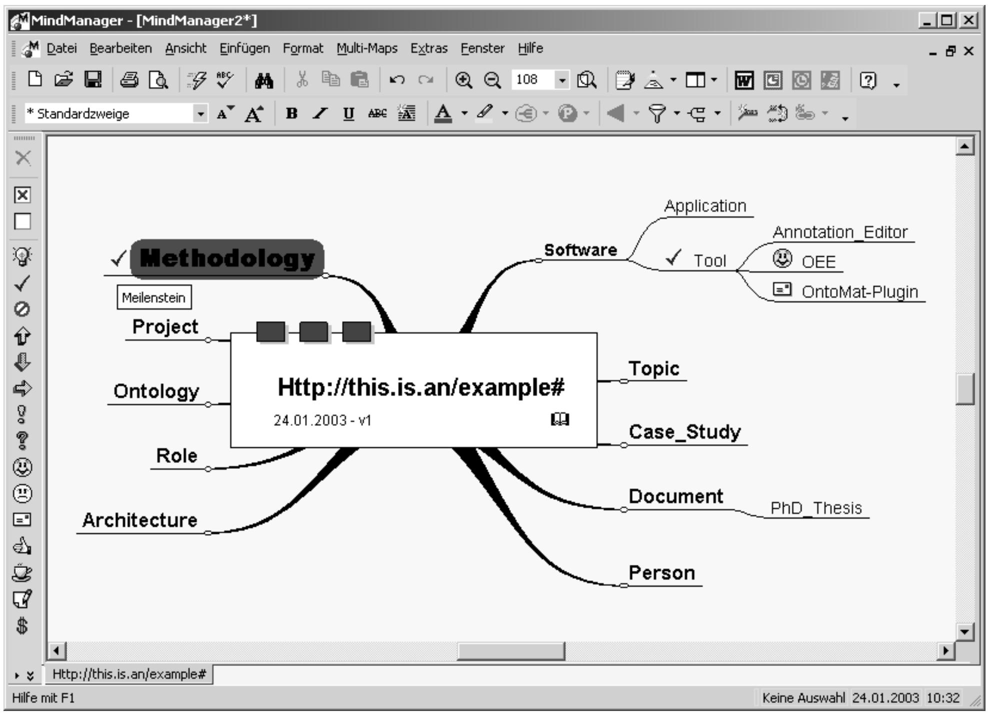
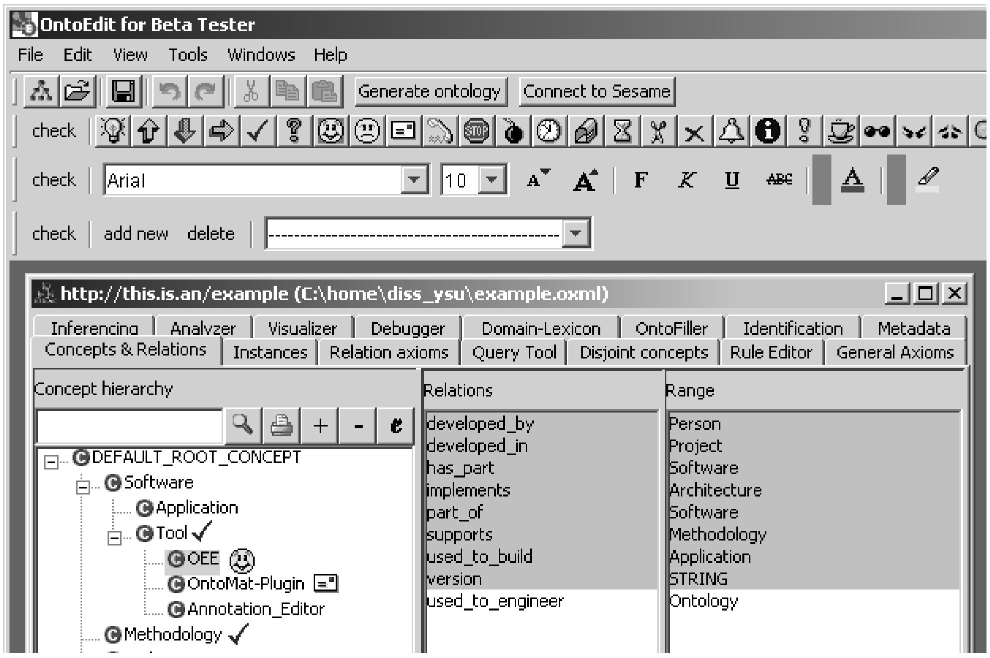


Figure 7.3: Example mind map about this work

The MindManager™ 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 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 about their 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. Further mind map™ examples are given in Figures 11.19 and 11.21.

The Mind2Onto framework integrates the mind map™ tool into the ontology engineering methodology. It consists of two parts, *viz.* (i) we developed scripts for the MindManager™ that import/export mind maps™ from/to the OXML format and provide additional needed functionalities such as creation of namespaces and unique identifiers, and (ii) similar to the MindManager™ we use layout information and icons, so-called "codes", to present concepts within OntoEdit. Figure 7.4 illustrates the second part of the Mind2Onto framework, *i.e.* the implementations in OntoEdit.



Relations	Range
developed_by	Person
developed_in	Project
has_part	Software
implements	Architecture
part_of	Software
supports	Methodology
used_to_build	Application
version	STRING
used_to_engineer	Ontology

Figure 7.4: Support discussions with Mind2Onto

We imported the mind map™ shown in Figure 7.3 into OntoEdit via the Mind2Onto import/export facilities. As mentioned above, each concept can be marked with codes, *e.g.* like the concept OEE is marked with a *smiley*. Each code has an assigned meaning, *e.g.* the *smiley* has attached the meaning "That is good!". Furthermore, similar to the MindManager™ but also other common applications, the font, font size, font type and font color can be changed for each concept. *E.g.* by enlarging concepts or changing the color to red, one might attract the attention to such concepts if further discussion about them is needed[2]. During the refinement phase (*cf.* Section 4.5) an ontology engineer can now add in OntoEdit further relationships, attributes, instances and axioms to the hierarchical structure.

To summarize the main advantages: (i) by assigning codes one might indicate which issues are to be discussed, and (ii) by changing the layout one might draw the attention to it, to facilitate discussions *e.g.* in large structures. The usage of codes and layout

[2]Editing font information sounds very simple and still useful, but currently no other OEE supports such a feature!

information in OntoEdit fulfills two purposes, *viz.* (i) to close the gap between the usage of the MindManager™ and OntoEdit and (ii) to improve the collaboration of ontology engineers that work on the same ontology at different times.

Outlook: Currently the layout information and codes can only be used for concepts. In the future we plan to extend that also for relations, instances and possibly axioms. Another open issue is the transformation of mind maps™ into ontologies. Currently this requires a manual effort of ontology engineers. In the future we envision a possible semi-automatic tool support by applying ontology learning methods (*cf.* (Maedche, 2002a)), *e.g.* to support the decision whether a branch in a mind map™ reflects a concept, a relation or an instance – or a even combination of them.

7.3 External Support

Note: External plugins (*i.e.* plugins that are developed outside the scope of this work, but extend OntoEdit similar to the methodology plugins) that support the methodology are only briefly mentioned. More detailed information can *e.g.* be found in (Ontoprise, 2002b).

Domain Entry Collection

The plugin enables users to expand the ontology by additional expressions for the concepts. Currently these expressions can be:

- synonyms (same meaning as a selected concept),
- domain entries (same domain as a selected concept), and
- classifications (documents belonging to a selected concept).

Often the kickoff phase starts with collecting relevant concepts for a given domain. They can be stored in the domain lexicon to serve *e.g.* as a starting point for the formalization during the refinement phase. Thus, they inherently support the collaboration between ontology engineers and domain experts. Domain lexicons can be exported into OXML or F-Logic format. A specialized import and export suits to the knowledge retrieval platform Semantic Miner (*cf.* (Moench, 2003) for further information).

8 Refinement Support

Overview

In **this chapter** we present specialized plugins that support the refinement phase of the Knowledge Meta Process. The OntoFiller plugin presented in Section 8.1 helps (i) to translate ontologies into multiple languages and (ii) to add documentations (in multiple languages). In Section 8.2 we present Client-Server Framework that enables the distributed engineering of ontologies. Both plugins provide support for **R3: Methodology** and **R4: Collaboration**. Last, but not least, we briefly introduce further support for the refinement phase by external plugins in Section 8.3.

Note: The ontology that is used as an illustrating example is included completely in Appendix D.

References: This chapter is mainly based on parts of (Sure et al., 2002a).

8.1 Documentation & Translation

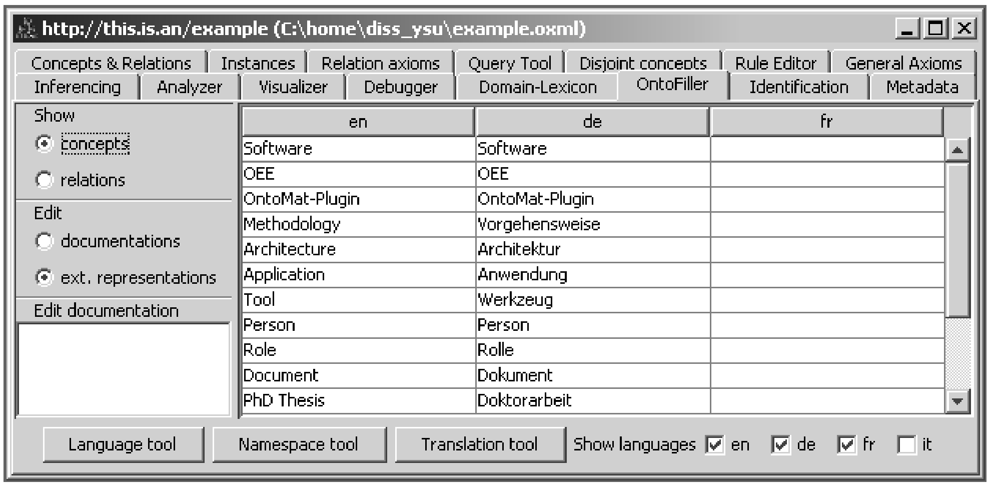
During the refinement phase of the Knowledge Meta Process (*cf.* Section 4.5) the semi-formal description of an ontology is refined into a target ontology. Typically one starts by creation of a taxonomical concept structure and adding of further relationships. However, an important aspect might be also the translation of concepts and relations of the ontology into different languages. Clearly, a well-documented ontology is more likely to be understood by people. Thus the documentation of concepts and relations should be integrated into the ontology engineering process. Both aspects are tackled by the OntoFiller plugin.

OntoFiller addresses a specific requirement from the Swiss Life case study (*cf.* Section 11.4), *viz.* the need for ontologies in multiple languages. 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). The OntoFiller plugin is characterized in Table 8.1.

Table 8.1: OntoFiller characterization

Methodology:	Refinement
Task:	Documentation and translation
Benefits:	User guidance
Requirements:	Methodology (R3) and Collaboration (R4)
Space:	Different place
Time:	Separated in time
Group:	Ontology engineers

As shown in Figure 8.1, 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 and documentations in various languages.



Show: [✓ concepts] relations

Edit: documentations [✓ ext. representations]

en	de	fr
Software	Software	
OEE	OEE	
OntoMat-Plugin	OntoMat-Plugin	
Methodology	Vorgehensweise	
Architecture	Architektur	
Application	Anwendung	
Tool	Werkzeug	
Person	Person	
Role	Rolle	
Document	Dokument	
PhD Thesis	Doktorarbeit	

Show languages ☑ en ☑ de ☑ fr ☐ it

Figure 8.1: Inserting multi-lingual external representations for concepts and relations with OntoFiller

Additionally, this plugin contains a translation support on top of the free German-English dictionary LEO[1]. 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 retrieved hints the appropriate translation for his purpose.

As a side effect this plugin allows several further functionalities, *viz.* (i) viewing and

[1]LEO dictionary, see `http://www.leo.org/`

editing all namespaces used in an ontology for concepts and relation and (ii) detecting all used languages in an ontology based on the language tags of XML.

Discussion: However, the translation of concepts into different languages is not without pitfalls. Our approach relies on the assumption that concepts and relations build the same structures in different languages. This assumption was valid for the Swiss Life case study (*cf.* Section 11.4), but in other scenarios this is not necessarily the case. For more complex ontologies is rather likely that the structures differ in multiple languages. Main reasons are different meanings of concepts (even for direct translations) and especially of relationships. To solve this problem, one would need to create for each language at least one separate ontology that reflects *e.g.* cultural backgrounds. To enable multi-lingual knowledge sharing one would define mappings between these ontologies. This is a quite immature research area with many open questions, first implementations exist (*cf.*, *e.g.*, Section 8.3) but are not yet applied to such problems.

Outlook: The translation support is planned to be extended also for other languages. The Altavista Babelfish[2] offers translations for a set of languages. It can be accessed via a SOAP web service interface.

8.2 Distributed Engineering

The Client-Server Framework allows members of an engineering team to collaborate even though they are geographically distributed and still modify the ontology at the same time. It is characterized in Table 8.2.

Table 8.2: Client-Server Framework characterization

Methodology:	Refinement
Task:	Locking and transaction management
Benefits:	Consistency and concurrency
	during distributed ontology development
Requirements:	Methodology (R3) and Collaboration (R4)
Space:	Different place
Time:	Same time
Group:	Ontology engineers

The description in this section is rather technical to show the practical implications

[2]Altavista Babelfish, see `http://babelfish.altavista.com/`

of this challenging approach. We have implemented[3] a client-server architecture (*cf.* Figure 8.2) in which the clients connect to an ontology server and can change or extend the ontology. All clients are immediately informed of modifications of other ontologists. Ontology engineers can store comments (*e.g.* explaining design decisions) in a documentation field for each concept and relation. By this way, one of the main features of ontologies, *i.e.* their consensual character, is supported. Collaborating ontologists must agree on the modeling decisions that are made. Therefore the possibility to monitor the development process of all collaborators is essential for reaching the goal of a shared ontology.

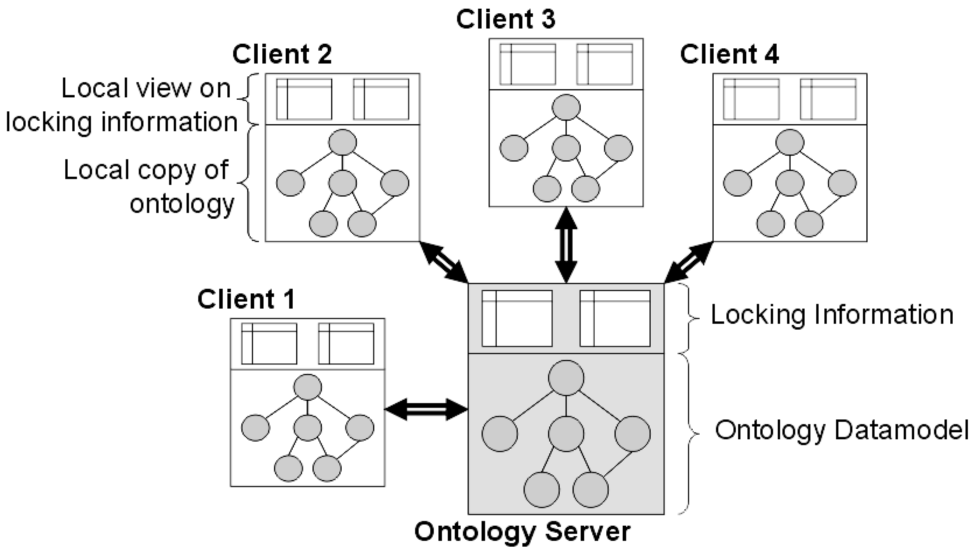


Figure 8.2: Client-Server architecture of OntoEdit

Transaction management[4]: In a distributed development environment certain mechanisms must be implemented to ensure safe development conditions, such as consistency of the models and the provision of a minimum degree of concurrency. To reach this goal we employed a locking and transaction protocol and implemented a distributed event model on the basis of Java-RMI (remote method invocation).

To guarantee consistent models the clients are forced to obtain locks for each resource (*e.g.* concept, instance, relation) that they want to modify (*e.g.* add a superconcept, add an attribute-value pair to an instance, or change the arity of a relation). The

[3]**Acknowledgements:** the Client-Server Framework, in particular it's implementation, was done by co-authors of (Sure et al., 2002a), *viz.* Michael Erdmann and Dirk Wenke, Ontoprise GmbH. Together we integrated it into the methodological framework.

[4]For introductory material on transactions and locking protocols we refer *e.g.* to (Bernstein et al., 1987; Gray & Reuter, 1993).

server denies the (write-) access to a resource if the resource is not locked by the client that attempts to modify it. Clients can obtain locks either by explicitly locking these resources, or more conveniently, by a begin of transaction (BOT) that is accompanied with a list of needed resources. If not all resources can be assigned to the calling client the BOT fails and the transaction is immediately aborted. Otherwise the server locks the needed resources for the client, so that no other client can manipulate them until the end of the transaction is reached. Now the client can manipulate the locked resources until it commits the transaction. After a commit all locked resources are freed again and the operations performed in the body of the transaction are actually applied to the datamodel. Afterwards, events are created to inform the other clients of the modifications performed. If the transaction needs to be aborted by the client all operations are undone, all locks are removed, and no events are fired.

Transactions may be nested to make complex operations possible without the need of rollback mechanisms. *E.g.* the datamodel procedure of moving a concept from one superconcept to another one consists of two sub-transactions (remove a superconcept-relationship to the first superconcept and establish a new one for the second concept) that must be performed all together or none at all. Because of the necessity of nested transactions we implemented a strict two phase locking protocol (S2PL). In this protocol additional resources can be achieved (and locked) within the body of a transaction. Our implementation of the S2PL allows for arbitrarily nested transactions. The execution of inner transactions and the release of all locked resources is postponed until the outermost commit or abort is finally reached. Again, only after the final commit events are sent to the other clients. We employ the S2PL because (i) it allows for nested transactions and (ii) prevents cascading aborts. Thus, clients can be immediately informed if a planned operation will commit or is prohibited due to unavailable resources. (iii) S2PL prevents also deadlocks since resources are only locked in a BOT if all locks can be achieved. Other locking protocols are either too inflexible (like conservative locking (C2PL) that cannot lock resources in addition to the locks of the BOT and thus, is not suitable for nested transactions) or provide chances of deadlocks that must be appropriately handled.

To reduce communication overhead, save bandwidth and because transactions are relatively short lived no information about transactions (esp. not about locked objects within a BOT) is communicated from the server to other clients, *i.e.* the local view on locking information within a client (*cf.* Figure 8.2) contains all resources that are locked by this client (by a BOT) but none that have been locked by a BOT of any other client. Nevertheless, another kind of locking information *is* distributed to all clients. An ontologist can lock a whole subtree of the concept hierarchy. The server informs all clients of this locking operation.

Locking subtrees of the concept hierarchy: A common practice in ontology engineering is to start with a top level structure and to refine it later on. Different parts of an ontology can be refined by different ontologists or groups. These collaborators should be able to work on their parts of the ontology with as few interference with other ontologists as possible. This is achieved in OntoEdit by the possibility of locking a complete subtree of the concept hierarchy. After the subtrees have been locked no conflicts can arise anymore, and what is equally important, the need to check for locking information *with the server* is reduced drastically. Since most modeling operations will occur within the scope of the subtrees, *i.e.* will mainly access already locked resources, the client can decide *locally* whether these operations are permitted or not.

This (tree-) locking information is distributed to all other clients and visually indicated in the GUI. Due to the distribution of this information clients can often check locally whether a transaction will be permitted or not. If all needed resources are marked as "locked by me" in the local view on the locking information (*cf.* Figure 8.2) a BOT can be safely accepted. If at least one resource is marked as being locked by another client the current client can definitively reject a BOT (or a lockSubTree request). Only if resources are requested in a BOT for which no information is locally available, the server has to be consulted.

What does locking a concept mean? Locking resources in relational databases (DB) means the DB administrators or application developers must decide whether to lock an attribute, a tuple, or a complete table (*i.e.* relation). Since the basic datamodel for ontologies is much richer (*esp.* due to hierarchical relationships between concepts, between relations, and between instances and concepts) the decision of what a lock entails is more complex.

The most simple answer would be to lock the complete ontology with all its components. But this solution is ruled out since it would disallow any kind of concurrency and distributed collaboration. Another simple answer would be to lock the resources that are to be modified within a transaction, *e.g.* the resource X in the transaction that states that concept X has a superconcept Y. Apparently, for this transaction concept Y should also be locked since a new subconcept for Y is defined. Thus, the second simple approach seems to lock too few resources.

Due to hierarchical relationships between concepts locking a concept X implies *read-locks* for all super-concepts of X. A read-lock marks a resource as being read-only, *i.e.* modifications to it are currently disallowed. If a read-lock for at least one superconcept cannot be achieved X will not be locked and the BOT fails. Thus, no operations may modify X. Read-locks can be yielded to multiple clients at the same time without conflict. If a client is the only one that read-locked a resource the client can achieve a stricter (write-)lock. Other clients cannot.

Figure 8.3 shows an example. Crosses mark concepts that are (read-)locked by other clients and may not be edited. Bullets mark concepts that are (write-)locked by the current user and may be edited, altered and removed at will. Concepts without additional icons are currently not locked by other users and therefore available for locking. *E.g.* user “1” (left screenshot) (write-)locked the concept Document and (as explained above) this implies a (write-) lock on the subconcept PhD Thesis. These concepts are now excluded from the list of available concepts for user “2” (right screenshot), therefore marked with a cross in the GUI of user “2”. Vice versa, user “2” (write-)locked the concept Tool and at the same time all it’s subconcepts. For user “1” this implies the exclusion of Tool, it’s subconcepts and superconcept, *viz.* Software, from the list of available concepts for locking.

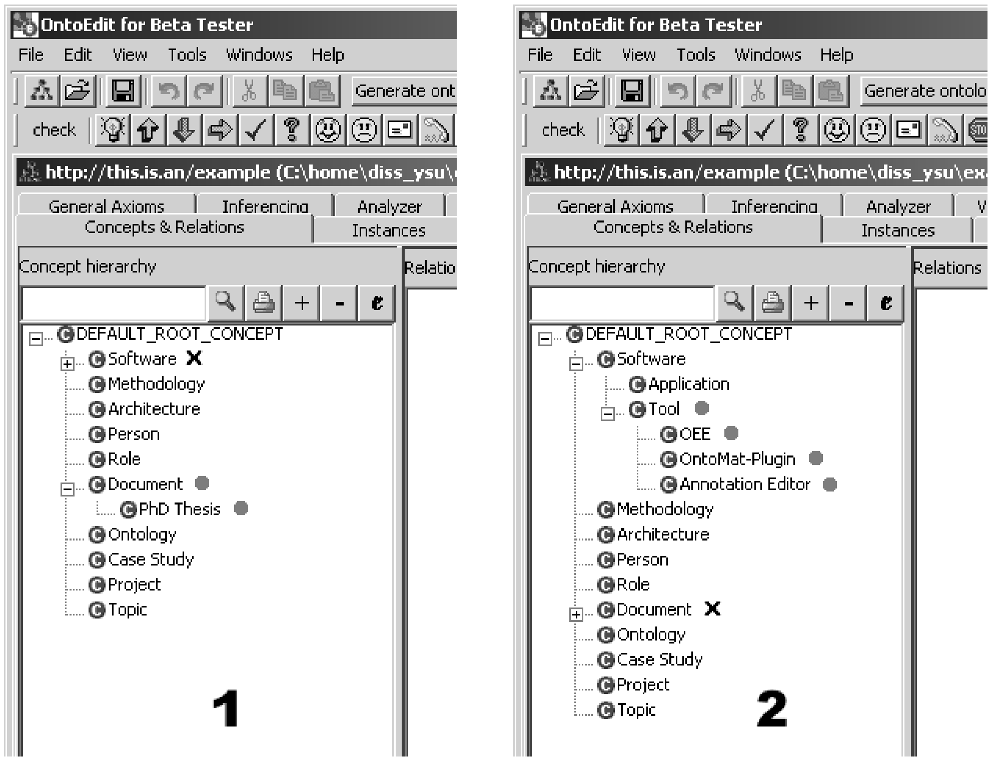


Figure 8.3: Locked trees in OntoEdit

The reason why a lock propagates from one resource to another in the ontology can be seen in the following example scenario: Assume, X is a subconcept of Y and Y has a relation A with range Y. Assume, we want to restrict the value range of A for X from Y to X. Thus, in the BOT we just lock the concept X and call the appropriate operation on X. Before we send the commit another client (after locking Y) changes the name of A to B and commits. If we now commit our transaction the semantics of the combined operations is not defined. Does X now have two independent attributes A and B? Or is attribute A totally lost as well as our newly defined range restriction?

Both situations are unsatisfactory. Thus, to prevent them superconcepts need to be read-locked.

Outlook: The transaction management is currently implemented for concepts and still in an early stage. However, the extension for locking relations, instances and especially for axioms is non-trivial and still requires a substantial amount of research to develop sophisticated locking strategies, otherwise one easily ends up in locking each time the whole ontology.

8.3 External Support

Note: External plugins (*i.e.* plugins that are developed outside the scope of this work, but extend OntoEdit similar to the methodology plugins) that support the methodology are only briefly mentioned. More detailed information can *e.g.* be found in (Ontoprise, 2002b).

Axioms Management

Axioms are quite difficult to model. OntoEdit offers several functionalities to create and manage axioms, to make the life of ontology engineers easier.

The General Axioms editor allows users to define all kinds of axioms. All existing axioms can be grouped into folders to structure them. "Typical axioms" (inverse, disjoint, *etc.*) are grouped together and stored in specialized folders to provide a better overview. Modelling such axioms is done in a "notepad"-like manner and the syntax of axioms has to be F-Logic (*cf.* (Kifer et al., 1995)).

There exists a newly Graphical Rule Editor plugin that allows for graphically oriented creation of axioms. Unlike the General Axioms plugin, one is assisted by a graphical interface, where no F-Logic knowledge is needed.

Generate Mapping Rules

The decentralized nature of the Semantic Web makes achieving consensus across communities difficult, therefore the generation of a single coherent ontology seems unrealistic. In order to balance the autonomy of each community with the need for interoperability, mapping mechanisms between distributed ontologies are required. A mapping framework for ontologies has *e.g.* been presented in (Maedche et al., 2002a).

The OntoMap plugin is an already existing plugin for generating arbitrary mappings between two ontologies. It allows for the mapping of data structures between ontologies via graphically oriented drag & drop. Mappings are stored as rules along with the mapped ontologies. They can be processed in Ontobroker during runtime.

9 Evaluation Support

Overview

In **this chapter** we present specialized plugins that support the evaluation phase of the Knowledge Meta Process. In Section 9.1 we show the OntoGenerator which aims at the creation of "synthetic ontologies" that allow for an evaluation of scalability of ontology based tools. The OntoAnalyzer presented in Section 9.2 helps in evaluating (formalized) guidelines to ensure that ontologies subscribe to predefined modelling guidelines. The OntoClean plugin shown in Section 9.3 implements the well-known OntoClean methodology for formal evaluations of ontologies based on philosophical principles. All three plugins heavily rely on the underlying Ontobroker inferencing engine and thus provide support for **R3: Methodology** and **R5: Inferencing**. Last, but not least, we briefly introduce further support for the evaluation phase by external plugins in Section 9.4.

Note: The ontology that is used as an illustrating example is included completely in Appendix D.

References: This chapter is mainly based on parts of (Sure et al., 2002b).

9.1 Scalability Evaluation

We address two of the mentioned evaluation aspects (*cf.* Section 4.6) for ontologies and related technologies: the technology-focussed evaluation and the ontology-focussed evaluation. The user-focussed evaluation was *e.g.* performed in two of the case studies (*cf.* Sections 11.5 and 11.6). Tool support for this evaluation was implemented outside of the ontology engineering environment and rather embedded into the ontology-based applications. This reflects the fact the the user perception of ontologies is rather dependant on the different usages of ontologies in applications.

The OntoGenerator plugin targets at technology-focussed evaluation, *viz.* scalability evaluation. The plugin is characterized in Table 9.1.

Table 9.1: OntoGenerator characterization

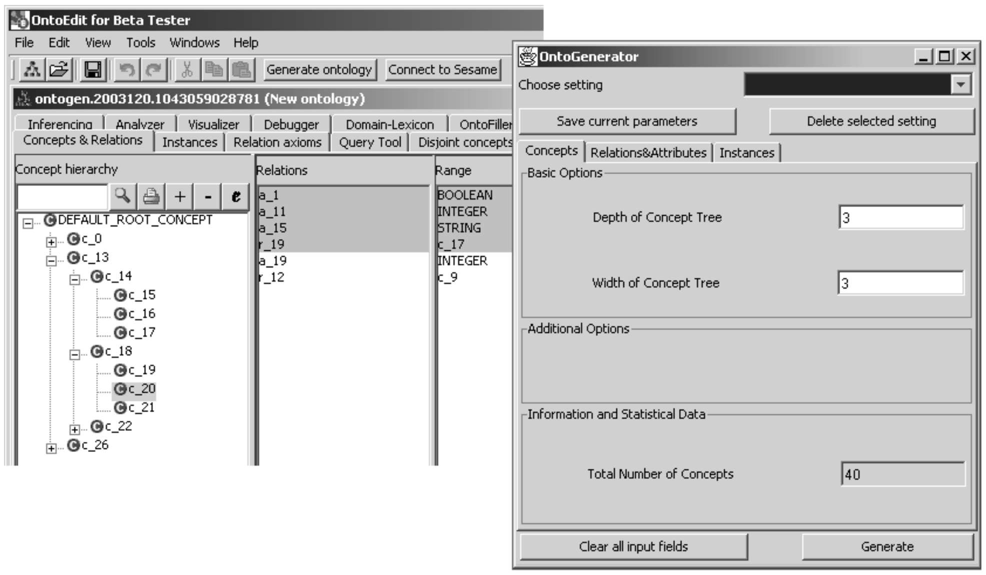
Methodology:	Evaluation (technology-oriented)
Task:	Generation of "synthetic" ontologies to support scalability evaluations
Benefits:	Modular extensibility, flexibly adaptable to different evaluation settings
Requirements:	Methodology (R3) and Inferencing (R5)
Axioms:	Application of predefined axioms

OntoGenerator extends OntoEdit by support for the generation of "synthetic ontologies". In contrast to well-known domain ontologies (*cf.* Section 3.2) they do not model a specific domain, *i.e.* the concepts, relations and instances created do not relate to specific objects in the real world but are artificially named entities that form ontology-like structures according to pre-defined parameters. They aim at supporting the evaluation and benchmarking of ontology based technologies. The consensual character for synthetic ontologies therefore does not lie in the agreement about a particular domain description but rather in the agreement what kind of model fits for the evaluation or benchmarking purpose. A synthetic ontology can be created with OntoGenerator according to various user-definable parameters. The plugin's modular design simplifies further extensions.

Modular design: As illustrated in Figure 9.1 OntoGenerator consists of a central plugin and several modular plugins for the generation of a specific section of an ontology. The central plugin manages the module plugins which do not directly interact with OntoEdit but only communicate with the central plugin. It also provides a mechanism for storing and re-loading parameter profiles. Each additional module is reflected in the GUI by an additional tab where parameters for the generation are captured. Currently there exist modules for (i) concepts, (ii) relations and attributes, and (iii) instances.

Parameters for concepts: With the current version of the concept-generating module, a symmetrical concept tree is generated. The user may enter values for depth and width of the concept tree to be generated. Parameter "depth" specifies the number of concepts that descend from the root concept in a direct line. "Width" indicates the number of subconcepts for each concept. At the bottom of the concept parameter input panel the total number of concepts that would be generated with the current parameter values is stated.

Parameters for relations and attributes: The module for the generation of relations and attributes can be customized by entering two parameter values, the total


OntoEdit for Beta Tester
File Edit View Tools Windows Help
Generate ontology
Connect to Sesame
ontogen.2003120.1043059028781 (New ontology)
Inferencing Analyzer Visualizer Debugger Domain-Lexicon OntoFiller
Concepts & Relations Instances Relation axioms Query Tool Disjoint concepts
Concept hierarchy
DEFAULT_ROOT_CONCEPT
c_0
c_13
c_14
c_15
c_16
c_17
c_18
c_19
c_20
c_21
c_22
c_26
OntoGenerator
Choose setting
Save current parameters
Delete selected setting
Concepts Relations&Attributes Instances
Basic Options
Additional Options
Information and Statistical Data
Clear all input fields
Generate

Relations	Range
a_1	BOOLEAN
a_11	INTEGER
a_15	STRING
r_19	c_17
a_19	INTEGER
r_12	c_9

Depth of Concept Tree: 3
Width of Concept Tree: 3
Total Number of Concepts: 40

Figure 9.1: Generating ontologies with OntoGenerator

numbers of attributes and relations. These are randomly attached to the concepts created by the concept module. For relations, the range concept as well is chosen at random.

Parameters for instances: Instances are only attached to the concept tree's leaves. The quantity of instances at each leaf is determined by several pre-defined distributions. Currently, the instance module features three distributions with different parameters:

- Random distribution: The only parameter for this distribution is the total number of instances that are to be created. These are spread randomly over all leaves.
- Linear distribution: Here the number of instances per leaf can be thought of as an increasing line. Both minimum and maximum number of instances can be chosen. The number of instances increases uniformly from “minimum number of instances” at the first leaf to “maximum number of instances” at the last leaf.
- Normal distribution: A normal or Gaussian distribution is determined by two parameters, expectation and standard deviation. Most leaves will have as many instances as specified by the expectation parameter. According to the standard deviation value, the number of leaves that have more or less instances attached, decreases symmetrically.

Discussion: Major benefits are (i) the modular extensibility of the plugin, so in future more advanced features can be added (see next paragraph) and (ii) the potential usage of all features the datamodel of OntoEdit has to offer to generate "synthetic ontologies" makes the plugin highly adaptable to different evaluation settings. A first usage scenario for OntoGenerator was *e.g.* the comparison of Ontobroker with XSB[1], a comparable inference engine ((Sure et al., 2002b)).

Outlook: For the future we plan to extend OntoGenerator with a module for rule generation on top. Potential issues for such an extension are (i) depth of rule trees, (ii) cyclicity of rules, (iii) length of rule cycles, (iv) complexity of rule bodies, and (v) transitivity of rules.

9.2 Guidelines Evaluation

Guidelines for ontology modelling help to ensure a coherent modelling of ontologies and thus to ensure a consistent level of quality. This affects multiple ontology engineers working on the same ontology, but also single or multiple ontology engineers working on multiple ontologies. On the one hand, support for evaluation therefore inherently enhances collaboration, respectively fulfills requirement **R4: Collaboration**. On the other hand, we make extensive use of inferencing capabilities by applying axioms to ontologies for evaluating them. We focus in this and the following section on the inferencing issues and therefore the support for requirement **R5: Inferencing**.

An integration of guideline checking into OEEs helps to evaluate the guidelines during modelling time and guarantees immediate feedback for ontology engineers. However, different requirements for ontologies, *e.g.* coming from envisioned applications that should be supported, might require different guidelines. Therefore, a flexible way of using and adapting guidelines is needed instead of hard coding them. Guidelines might be used for technology-focussed evaluations, *e.g.* to ensure that naming conventions are fulfilled (for instance, some inference engines do not allow for white spaces in concept identifiers while others allow for them as long as brackets enclose the identifiers *etc.*), or for ontology-focussed evaluations, *e.g.* as shown in the next section on formal evaluations of ontologies.

The OntoAnalyzer plugin offers flexible and modularized checking of formalized guidelines and constraints. The OntoAnalyzer is characterized in Table 9.2.

From our own experiences of ontology development and deployment we learned that for different purposes ontologies must have different properties, *e.g.* for different target applications (*cf.*, *e.g.*, Sections 11.4, 11.5 and 11.6). The definition of evaluation

[1]XSB is a "Logic Programming and Deductive Database system for Unix and Windows", see `http://xsb.sourceforge.net/`

Table 9.2: OntoAnalyzer characterization

Methodology:	Evaluation (ontology-oriented)
Task:	Checking of guidelines and constraints
Benefits:	Integrated and modular, thus adaptable, checking
Requirements:	Methodology (R3) and Inferencing (R5)
Axioms:	Application of predefined axioms

methods for such properties must be very flexible and easily maintainable. So it is not convenient to program it into the OEE itself, but rather to have a modular and flexible way to ensure the quality of an ontology by checking such properties. Logic is a very comfortable and powerful way on a conceptual level to express constraints for an ontology or to examine properties of an ontology. For that purpose the rule or constraint language must be able to access the ontology itself, *i.e.* to make statements about classes, relations, subclasses *etc.*

Formalization: F-logic allows to define statements and rules about the ontology (concepts, subconcepts, relations). *E.g.* the examination whether in an ontology *a concept has at maximum one super concept* may be expressed by the following "check" axiom:

```
FORALL C check("concept has more than one super concept",C)
    <- EXISTS S1,S2
    C::S1
    AND C::S2
    AND NOT equal(S1,S2).
```

Further examples for modelling guidelines can be derived from (Noy & McGuinness, 2001). *E.g.* consider the two guidelines ("slot" is synonymic used to our usage "relation"):

> *"If a list of classes defining a range of a slot includes a class and its subclass, remove the subclass."*
>
> *"If a list of classes defining a domain of a slot includes a class and its subclass, remove the subclass."*
> (Noy & McGuinness, 2001)

They can be formalized as the following F-Logic axioms and automatically checked within OntoEdit[2]:

```
FORALL C check("Remove concept from a range",C)
    <- EXISTS B,Domain,Rel
    Domain[Rel=>>B]
    AND Domain[Rel=>>C]
    AND C::B.

FORALL C check("Remove concept from a domain",C)
    <- EXISTS B,Range,Rel
    B[Rel=>>Range]
    AND C[Rel=>>Range]
    AND C::B.
```

How does it work: OntoAnalyzer is a tool which applies such axioms to an ontology opened in OntoEdit. The plugin transfers the ontology and check axioms to the Ontobroker inference engine and retrieves back and displays the results of this examination. OntoAnalyzer is able to load different axiom packages, each intended for a different target tool or target project.

To actually execute the checks, OntoAnalyzer asks for all values of the predicate `check(X,Y)`, as shown in the following:

```
FORALL X,Y <- check(X,Y).
```

The results are presented in the GUI of OntoAnalyzer.

Outlook: For the future we plan to develop standard rule packages for evaluation of various purposes to support efficient and effective engineering of ontologies and to improve the quality of ontologies at the same time. To enhance the usability of the plugin, the results derived from Ontobroker should immediately allow for the proposed actions, *e.g.* if a concept should be removed from the range of a relation, the dialog for doing so should be automatically opened.

9.3 Formal Evaluation

The previously introduced technology for checking ontologies can be extended to cover also formal ontology evaluations such as proposed by the OntoClean methodology (*cf.*, *e.g.*, (Guarino & Welty, 2000b; 2000c; Welty & Guarino, 2001; Guarino & Welty, 2002)).

[2]Please note that it is quite easy to present also the relation to which the domain and range concepts belong by extending the `check` predicate with an additional value.

The following building blocks constitute the basic infrastructure for implementing OntoClean: (i) a set of axioms that formalize definitions, constraints and guidelines given in OntoClean and (ii) a "meta-ontology", *viz.* the so-called "taxonomy of properties", that provides a frame of reference for evaluations. An ontology can be compared *vs.* a predefined ideal taxonomical structure to detect inconsistencies. Thus, the integration of the OntoClean methodology into OEEs enables an integrated quality control for ontologies. The OntoClean plugin makes use of and extends the basic infrastructure of the OntoAnalyzer (*cf.* previous Section). The plugin is characterized in Table 9.3.

Table 9.3: OntoClean Plugin characterization

Methodology:	Evaluation (ontology-oriented)
Task:	Formal evaluation of ontologies
Benefits:	Enhanced quality of concept hierarchies
Requirements:	Methodology (R3) and Inferencing (R5)
Axioms:	Application of predefined axioms

The OntoClean methodology is based on philosophical notions for a formal evaluation of taxonomical structures. 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., 2002b)). Core to the methodology are the four fundamental ontological notions of *rigidity*, *unity*, *identity*, 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, 2000a)). We can only briefly sketch the methodology in a simplified way and mention two of the introduced philosophical notions, *viz.* rigidity and unity:

- **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 it 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 (∼R) if and only if it is not essential to all its instances.

  Example: Consider for example the property of *being hard.* We may say that it is an essential property of hammers, but not of sponges. Some sponges (dry ones) are hard, and some particular sponge may be hard for its entire existence, however this does not make being hard an essential property of that sponge. The fact is that it *could have* been soft at some time, it just happened that it never was.

Furthermore, *being a person* is usually conceptualized as rigid, while, as shown above, *being hard* is not. Rigidity is a subtle notion: every entity that *can* exhibit the property *must* exhibit it. So, every entity that is a person must be a person, and there are no entities that can be a person but aren't.

The property *being a student* is typically anti-rigid – *every* instance of student is not essentially a student (*i.e.* may also be a non-student).

- **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 (∼U) if all its instances can possibly be non-wholes.

  Example: *E.g.*, the enterprize British Airways is a whole unified by the relation *has president*. To generalize, an *enterprize with president* carries unity since the relation *has president* is the relation that unifies every instance.

Based on these meta-relations OntoClean classifies concepts into categories as shown in Figure 9.2 (the figure is taken from (Welty & Guarino, 2001)). *E.g.*, a concept that is tagged with "+O +I +R" is called a "Type".

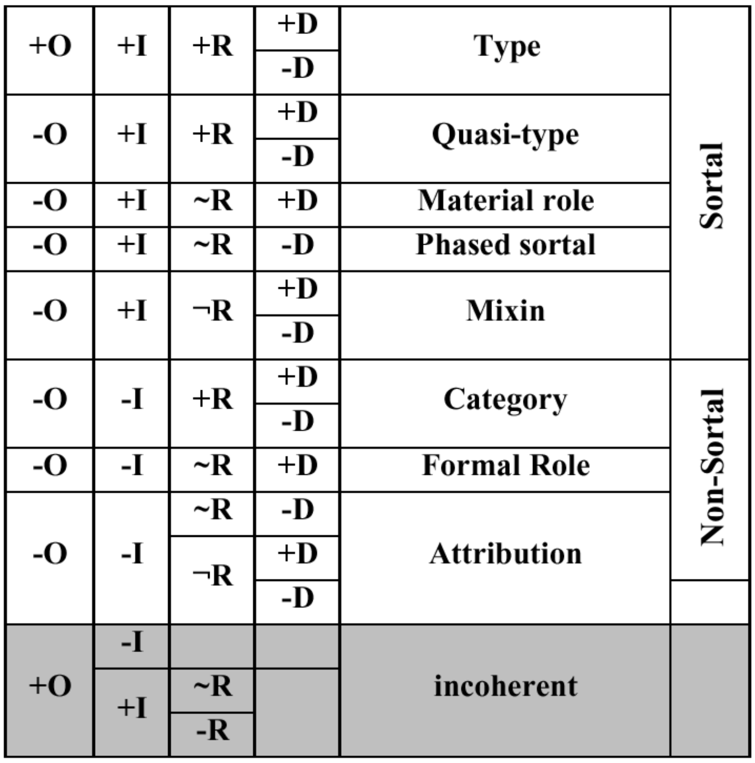
The aim of the methodology is to produce a "clean" taxonomy as shown in the ideal structure in Figure 9.3 (figure is taken from (Welty & Guarino, 2001)).

Beside these meta-relations OntoClean contains axioms that can be applied to evaluate the correctness of a given taxonomy. For instance, an axiom 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. As an example we present the formalization of the disjointness in F-Logic:

```
FORALL C check("A property cannot carry +R and -R",C)
   <- C[carryR->>"true"]
   AND C[carryNotR->>"true"].
```

Another example is, that a property that is defined as "anti-rigid" cannot subsume a property that is "rigid" (the check message is abbreviated for means of simplicity):

```
FORALL B ( check("~R can't subsume +R",B) )
   <- EXISTS C
   C::B
   AND B[antiR->>"true"]
   AND C[carryR->>"true"].
```



+O	+I	+R	+D	Type	Sortal
			-D		
-O	+I	+R	+D	Quasi-type	
			-D		
-O	+I	~R	+D	Material role	
-O	+I	~R	-D	Phased sortal	
-O	+I	¬R	+D	Mixin	
			-D		
-O	-I	+R	+D	Category	Non-Sortal
			-D		
-O	-I	~R	+D	Formal Role	
-O	-I	~R	-D	Attribution	
		¬R	+D		
			-D		
+O	-I			incoherent	
	+I	~R			
		-R			

Figure 9.2: Combinations of OntoClean meta-relations

To implement the OntoClean methodology in OntoEdit[3], we (i) formalized the constraints and definitions as axioms, and (ii) formalized the meta-relations and classifications as a "meta ontology" that can be used to classify concepts of an ontology.

We modelled the "meta ontology" and an example ontology (taken from (Welty & Guarino, 2001)) that has to be evaluated, in OntoEdit. Each concept of the example ontology, *i.e.* all subconcepts of the root concept of the example ontology, *viz.* "Entity", is then specified as being also an instance of the top-level concept "Property" of the meta ontology through an axiom:

```
FORALL C C:Property
    <- C::Entity.
```

[3]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 methodology with the process oriented METHONTOLOGY (Fernández-López et al., 1999) by extending the WebODE (Arpírez et al., 2001) ontology development environment (*cf.* `http://www.ontoweb.org/workshop/ontoweb2/slides/ontocleansig3.pdf`)

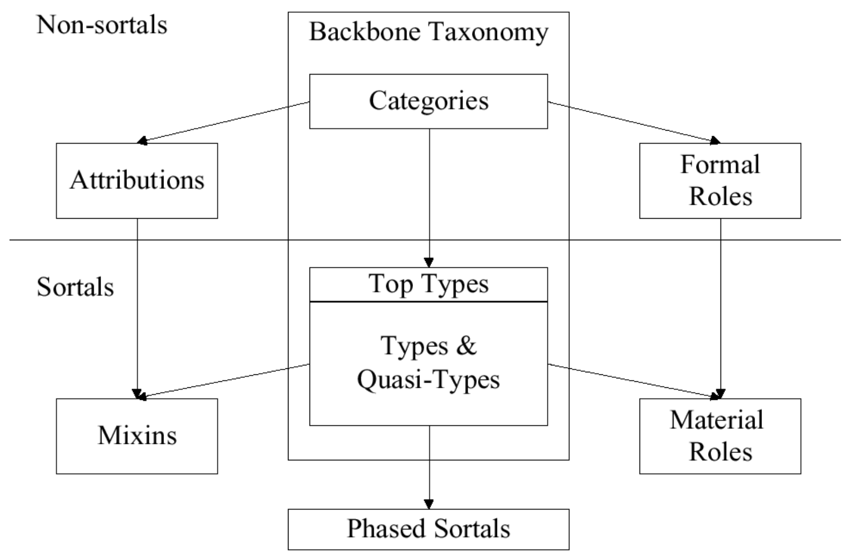
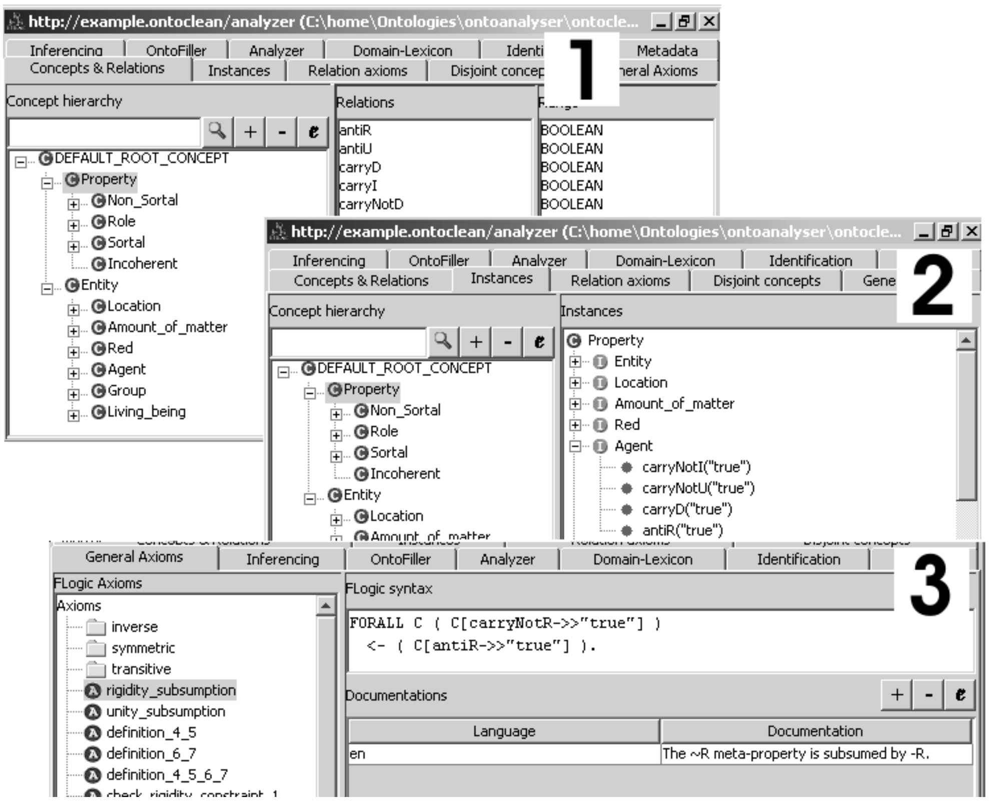


Figure 9.3: Ideal taxonomy structure

Figure 9.4 shows the subsequent steps during implementation and employment of OntoClean as a plugin in OntoEdit (the numbers in the figure correspond to the following enumeration):

1. model both ontologies, the taxonomy of properties and the example ontology,
2. fill the meta-relations with values (*i.e.* tag the concepts of the example ontology with "carryR" (+R) *etc.*), and
3. specify the definitions and constraints from OntoClean as axioms (here by using the General Axiom Editor, *cf.* Section 8.3). One can now ask queries to find inconsistencies in an ontology according to the OntoClean methodology.

Figure 9.5 shows the inconsistencies derived from applying the OntoClean axioms the example ontology by using the Inferencing plugin (*cf.* Section 9.4. On the left side the list of implemented axioms is shown, for testing purposes they can be switched on and off. On the right side the result from an evaluation is shown. *E.g.* the concept Agent is defined as "anti-rigid" and subsumes the concept Animal which is defined as "rigid". According to the OntoClean methodology this is a violation of a given constraint. To enhance the quality of the taxonomical structure an ontology engineer can now reconsider the modelled hierarchy.



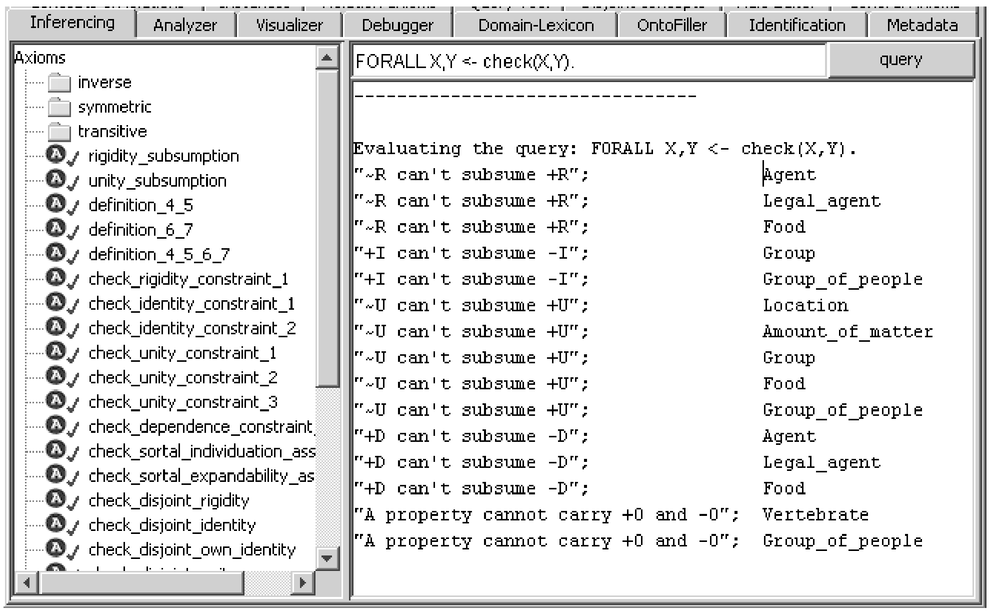
```
FORALL C ( C[carryNotR->>"true"] )
  <- ( C[antiR->>"true"] ).
```

Language	Documentation
en	The ~R meta-property is subsumed by -R.

Figure 9.4: Implementation of OntoClean in OntoEdit

Discussion & Outlook: The shown implementation is a first proof of concept. The next version of the plugin encapsules the meta ontology by using a dynamically built GUI to handle the tagging of concepts with meta-relations more intuitively. Similar to the OntoAnalyzer, the results should automatically guide users through a set of possible actions that can be performed to fix the detected inconsistencies.

However, the application of the OntoClean methodology requires significant training, only few people are currently able to apply it properly. In collaboration with the group from the Artificial Intelligence Laboratory of the Technical University of Madrid (UPM) and the inventors of the methodology we are therefore planning to implement a more user-friendly and intuitive solution.



```
-------------------------------

Evaluating the query: FORALL X,Y <- check(X,Y).
"~R can't subsume +R";                  Agent
"~R can't subsume +R";                  Legal_agent
"~R can't subsume +R";                  Food
"+I can't subsume -I";                  Group
"+I can't subsume -I";                  Group_of_people
"~U can't subsume +U";                  Location
"~U can't subsume +U";                  Amount_of_matter
"~U can't subsume +U";                  Group
"~U can't subsume +U";                  Food
"~U can't subsume +U";                  Group_of_people
"+D can't subsume -D";                  Agent
"+D can't subsume -D";                  Legal_agent
"+D can't subsume -D";                  Food
"A property cannot carry +O and -O";    Vertebrate
"A property cannot carry +O and -O";    Group_of_people
```

Figure 9.5: Deriving inconsistencies with the OntoClean Plugin

9.4 External Support

Note: External plugins (*i.e.* plugins that are developed outside the scope of this work, but extend OntoEdit similar to the methodology plugins) that support the methodology are only briefly mentioned. More detailed information can *e.g.* be found in (Ontoprise, 2002b).

Visualization (of Ontologies)

Visualization of ontologies facilitates the engineering tasks in general. Ontology engineers can use such visualizations *e.g.* to “get a feeling” for ontologies, especially for the complex structures created by relationships between concepts. Typically, OEEs provide some kind of “explorer”-like visualization of the concept hierarchies. Further visualizations are needed to represent *e.g.* relationships other than “is-a” taxonomies. Such visualizations may be reused to provide navigational structures for portals that are build on top of an ontology. Therefore the integration of such visualizations into OEEs helps to evaluate such structures before their implementation in a running application.

The Visualizer plugin (shown in Figure 9.6) enables such visualizations and allows the user to browse and edit the ontology by a graphical means. Concepts and instances are shown as blue and green bullets, which can be used as navigators either by double-click or right-click. The number of existing instances to a selected concept is shown in a small red square. To edit an ontology, one has to select the 'Edit' radio button. The right click menu on the bullets shows you the options for editing the ontology.

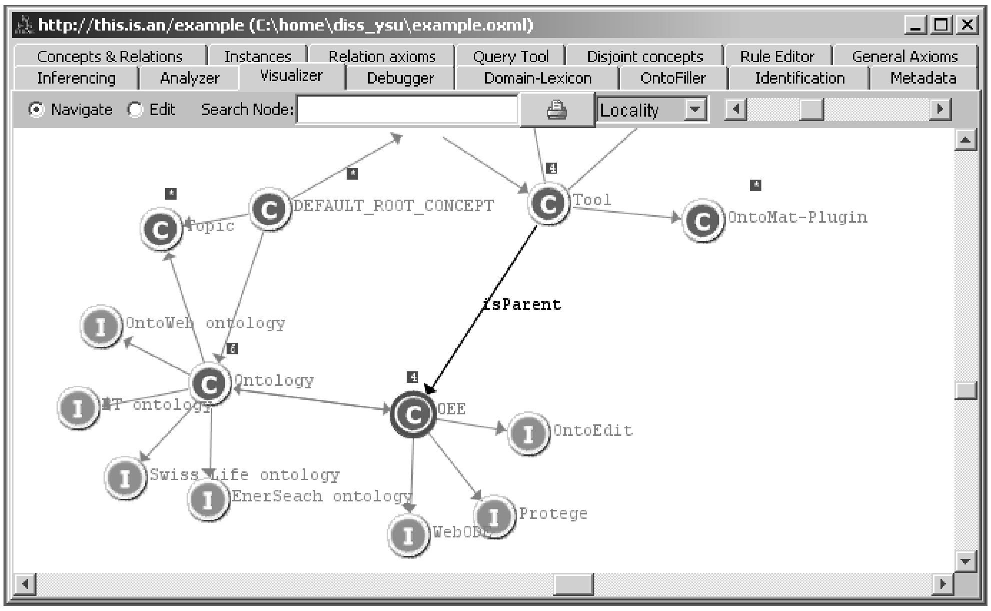


Figure 9.6: Visualizing ontologies in the Visualizer Plugin

Query Evaluation

The Inferencing plugin can be used to test the ontology and its axioms. Queries in F-Logic syntax can be tested within OntoEdit. All axioms can be switched on and off during runtime, thereby *e.g.* allowing for checking side-effects of axioms.

Graphical Query Evaluation

The helps to create queries for knowledge bases. It enables the user to generate the queries graphically based on one of the available ontologies in OntoEdit. There exist several possibilities for defining mappings such as

- concepts on concepts,
- attributes on concepts,

- attributes on attributes, and
- relations on relations.

Debugging of Axioms

The Rule Debugger plugin allows for a graphically oriented debugging of axioms. Therefore the evaluation of axioms is greatly facilitated.

10 Application & Evolution Support

Overview

In **this chapter** we present specialized plugins that support the application & evolution phase of the Knowledge Meta Process. In Section 8.2 we present the Sesame Plugin that enables storage and versioning of ontologies based on the Ontology Middleware Module which belongs to the Sesame RDF(S) repository. The plugin provides support for **R3: Methodology** and **R4: Collaboration**. Last, but not least, we briefly introduce in Section 10.2 further external support for evaluation that is also based on the flexible OntoMat plugin framework.

References: This chapter is mainly based on parts of (Sure & Studer, 2002c).

10.1 Storage & Versioning

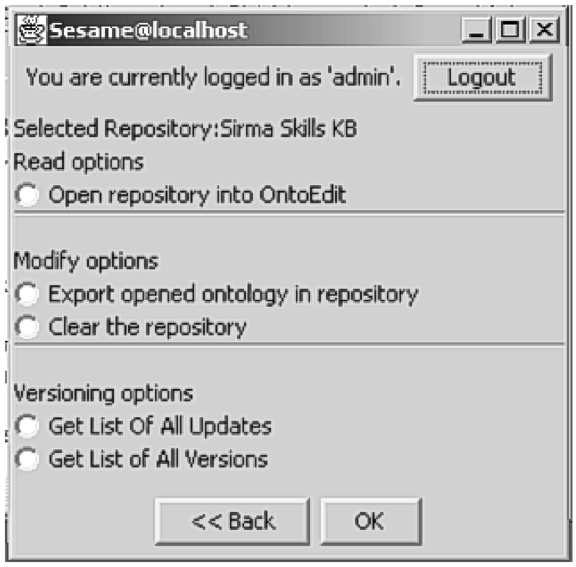
The Sesame Plugin is characterized in Table 10.1.

Table 10.1: Sesame Plugin characterization

Methodology:	Application & Evolution
Task:	Storage, change management
Benefits:	Persistent storage of multiple versions
Requirements:	Methodology (R3) and Collaboration (R4)
Space:	Different place
Time:	Separated in time
Group:	Ontology engineers

The Sesame Plugin connects OntoEdit to Sesame (*cf.* Section 11.2 or (Broekstra et al., 2002)). Sesame provides (i) a storage facility for RDF(S) ontologies, (ii) a query engine on top of the storage facility that makes Sesame also suitable as a backend for applications, and (iii) the Ontology Middleware Module on top of the query engine that supports the evolution of ontologies (*cf.* Section 4.7). It is noteworthy that Sesame can only handle RDF(S) ontologies and RDF data statements, but an extension for more sophisticated support for DAML+OIL is planned.

The plugin allows for connecting Sesame repositories via the protocols HTTP, RMI and SOAP. Figure 10.1 shows possible actions, *viz.* (i) Ontologies that are stored in Sesame can be opened in OntoEdit, (ii) ontologies that are opened in OntoEdit can be stored in Sesame, (iii) repositories in Sesame can be cleared, (iv) the list of all available updates of an ontologies can be retrieved (and a selected update can be opened in OntoEdit), and (v) the list of all available versions of an ontology can be retrieved (and a selected version of an ontology can be opened in OntoEdit).



You are currently logged in as 'admin'.

Selected Repository:Sirma Skills KB

Read options

Open repository into OntoEdit

Modify options

Export opened ontology in repository

Clear the repository

Versioning options

Get List Of All Updates

Get List of All Versions

Figure 10.1: Storing and versioning ontologies with the Sesame Plugin

Outlook: Sesame is being developed as open source. Currently it supports RDF(S), but future extensions are planned to cover also OWL specific features which would result in an update of the Sesame plugin to reflect the new functionalities.

10.2 External Support

Evolution of ontologies is still an immature area. But, when ontologies grow in size, the complexity of change management increases, and rather simple strategies as presented in the previous section are not sufficient any longer. Therefore, more advanced evolution strategies are needed. (Stojanovic et al., 2002a) presents an evolution strategy that encapsulates policies for evolution with respect to user's requirements. OntoMat–SOEP, the tool that implements the strategy, is based – like all previously presented OntoEdit plugins – on the OntoMat plugin framework. However, the internal data structures are aligned to the KAON framework (*cf.* (Motik et al., 2002; Bozsak et al., 2002)). For the future it is planned to align the data structures of OntoEdit and KAON to make the OntoMat plugins more reusable.

Part IV

Case Studies

*“**dogfood** n.*

*[Microsoft, Netscape] Interim software used internally for testing. **‘To eat one’s own dogfood’** (from which the slang noun derives) means to use the software one is developing, as part of one’s everyday development environment (the phrase is used outside Microsoft and Netscape).* [. . .]”

— http://www.tuxedo.org/˜esr/jargon/html/entry/dogfood.html

11 On-To-Knowledge

In **this part** we present several **case studies** that how the methodology and tools presented in the parts before are applied in different scenarios. On the one hand, three case studies of the On-To-Knowledge project shown in Chapter 11 represent a broad spectrum of use cases in corporate intranets. On the other hand, the Semantic Portal of the thematic network OntoWeb shown in Chapter 12 illustrates the application of the presented work in the World Wide Web. All case studies provide real world experiences and show different aspects of ontology based knowledge management applications. Additionally, we performed as part of OntoWeb an evaluation of ontology based tools, *viz.* a comparison of different Ontology Engineering Environments.

Overview

In **this chapter** we start by introducing the project On-To-Knowledge and describing the contribution of this work to the project in Section 11.1. We continue with the introduction of the technical architecture in Section 11.2 and give an overview on how the tools are used within three different case studies in Section 11.3. Subsequently we present each case study, *viz.* (i) "Skills Management" at Swiss Life in Section 11.4, (ii) "Communities of Knowledge Sharing" at BT in Section 11.5 and (iii) "Virtual Organization" at EnerSearch in Section 11.6. We conclude with presenting the main lessons learned during the case studies in Section 11.7.

References: This chapter is mainly based on (Sure et al., 2003a), (Sure, 2002b), (Lau & Sure, 2002), (Davies et al., 2003) and (Sure & Iosif, 2002).

11.1 About the Project

The EU IST-1999-10132 “On-To-Knowledge: Content-driven Knowledge Management through Evolving Ontologies” (OTK) project[1] (*cf.*, *e.g.*, (Davies et al., 2002b)) is part of the 5th framework of the Information Societies Technologies (IST) Programme of the European Union (EU). Within the project one aims at developing and exploring sophisticated methods and tools for knowledge management as described in the OTK Annex.

> *“The On-To-Knowledge project applies ontologies to electronically available information to improve the quality of knowledge management in large and distributed organisations. . . . we will develop a methodology and tools for intelligent access to large volumes of semistructured and textual information sources in intra-, extra-, and internet-based environments to employ the full power of ontologies in supporting knowledge management . . . The goal of the On-To-Knowledge project is to support efficient and effective knowledge management.”*
> (On-To-Knowledge, 1999)

OTK partners include (i) **technology providers** who build an ontology based tool environment to support the acquisition, maintenance and access of information, (ii) **technology users** who evaluate and use the tool environment in industrial case studies and (iii) a **methodology provider** who guides and assists during the application of the tool environment in the case studies.

The **tool suite** developed by the technology providers consists of a set of tools for different purposes, *e.g.* access, extraction and storage. The tools are in principle stand-alone solutions that are integrated via the agreement on OIL, the “Ontology Inference Layer”, as a common interchange language for ontologies and metadata. In the case studies the existing tools are bundled together and used in different configurations to meet the different requirements each case study has.

Three **case studies** are carried out by the technology users 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. The three case studies are (i) “Skills Management” at the company Swiss Life, (ii) “Communities of Knowledge Sharing” at the company BT and (iii) “Virtual Organization” at the company EnerSearch.

[1]http://www.ontoknowledge.org

Within the project the **On-To-Knowledge Methodology** developed by the methodology provider is the glue between technology providers and technology users. The On-To-Knowledge Methodology sticks all parts together and provides a general overview of all parts. The methodology provider captures lessons learned from the OTK case studies while applying the OTK tool set.

The following items show the relationship of work performed in the project and parts of this thesis:

- The On-To-Knowledge Methodology (*cf.* (Staab et al., 2001; Sure & Studer, 2002a)[2]) was initially developed in the project and refined in this thesis in Part II.
- OntoEdit and the plugins described in Part III were developed as part of the On-To-Knowledge tool suite (*cf.* (Sure et al., 2002a; 2002b)[3]). Most of the requirements for the plugins were derived from the On-To-Knowledge case studies.
- The three case studies were carried out in close cooperation between the methodology provider and the technology users (*cf.* (Lau & Sure, 2002; Davies et al., 2003; Sure & Iosif, 2002)). Thus, the case studies served as an evaluation not only for the tool suite, but also for the On-To-Knowledge Methodology. The following Sections 11.4, 11.5 and 11.6 about the case studies reflect the work performed in close cooperation with the case study providers. Further information about each case study can be found in the corresponding On-To-Knowledge deliverables which are mentioned at the beginning of each section.
- As illustrated in Section 11.3, the case studies explored different configurations of the tool suite. However, during the final review meeting[4] an integrated life demo was presented to show all tools interacting with each other[5].

11.2 Technical Architecture

This section about the technical architecture summarizes the work performed on the technical architecture in the On-To-Knowledge project. As such, it reflects the joint

[2]The evolvement of the methodology and the accompanying tool support can be recaptured in the corresponding On-To-Knowledge deliverables: (Schnurr et al., 2000; Sure & Studer, 2001b; 2002b; 2002c).

[3]The following On-To-Knowledge deliverables cover OntoEdit and plugins related issues: (Sure & Studer, 2001a) and (Sure & Studer, 2002c).

[4]Final On-To-Knowledge project review meeting, see `http://www.ontoknowledge.org/final-review.shtml`

[5]OTK Tool Suite Demo, edited by Y. Sure, slides available for download at `http://www.ontoknowledge.org/downl/finalreview-demo.pdf`

work of numerous project partners. Firstly, we briefly introduce all tools that were developed as part of the project. Secondly, we give a summary about the ontology language OIL which was also developed as part of the project.

11.2.1 OTK Tool Suite

A key outcome of the On-To-Knowledge project is the resulting "OTK Tool Suite" [6]. Several consortium partners are participating in the effort to realize in software the underpinning ideas and theoretical foundations of the project.

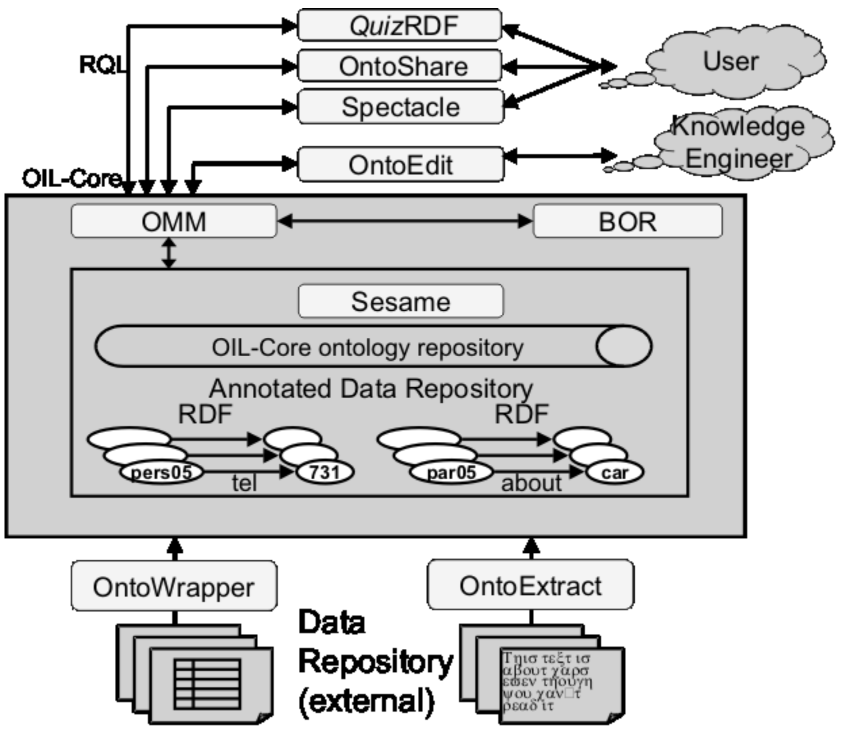


Figure 11.1: OTK technical architecture

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 11.1). 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

[6]The technical requirements to run the tool suite can be found in the OTK Technical Factsheet, *cf.* (Sure, 2002a), available for download at
`http://www.ontoknowledge.org/downl/OTK-Factsheet.pdf`

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 11.2.2).

*Quiz*RDF: Full Text Searching plus RDF Querying

*Quiz*RDF[7] (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. *Quiz*RDF 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.

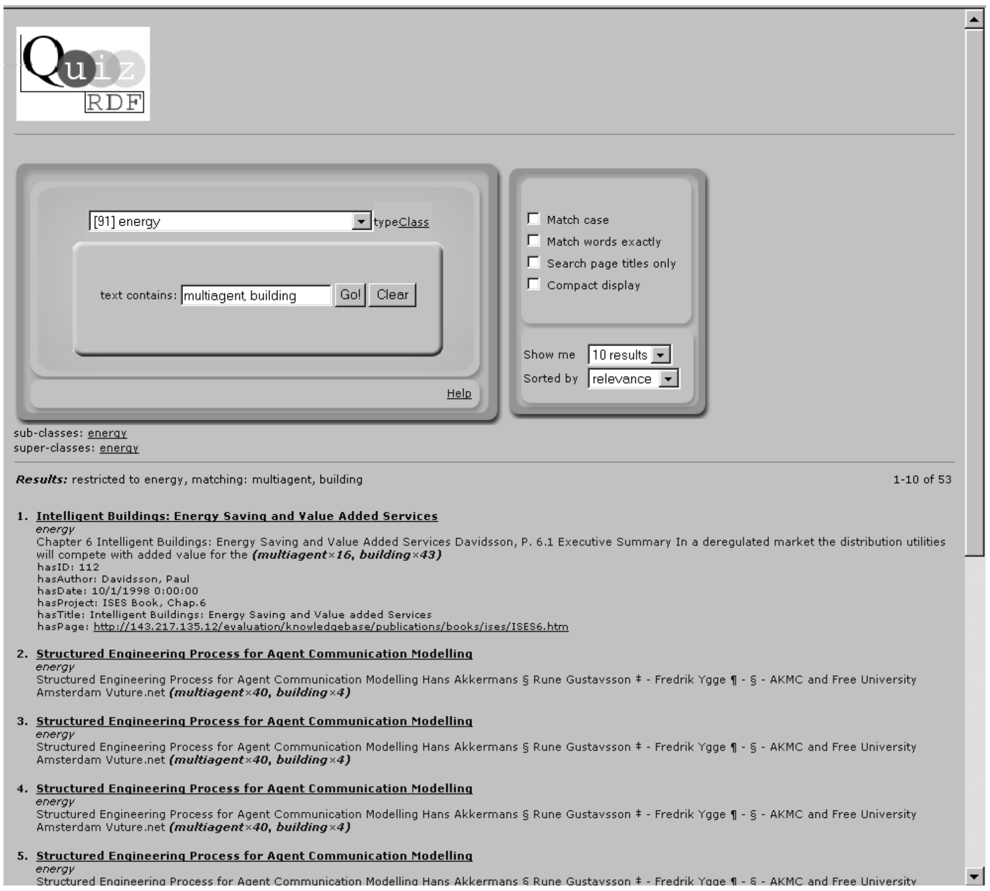
However, *Quiz*RDF's indexing and retrieval technique is also designed to use domain knowledge that is made available in the form of ontologies specified as OIL core. RDF resources are Web pages or (parts thereof) and such pages or segments are effectively ontological instances. Correspondingly, resource types are ontological classes. The information items processed by *Quiz*RDF are RDF resources. During indexing *Quiz*RDF assigns content descriptors to RDF resources. Content descriptors of a resource are terms (words and phrases) that *Quiz*RDF obtains from a full text analysis of the resource content and from processing all literal values that are directly related by a property. They also retain structural information about the ontology.

In *Quiz*RDF the user can select from a list of all the resource types stored in the index. When searching by selecting a resource type, *Quiz*RDF 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. *Quiz*RDF 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 11.2 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

[7]Due to legal matters the formally known RDF*ferret* is now being called *Quiz*RDF.



sub-classes: energy
super-classes: energy

Results: restricted to energy, matching: multiagent, building 1-10 of 53

1. **Intelligent Buildings: Energy Saving and Value Added Services**
   energy
   Chapter 6 Intelligent Buildings: Energy Saving and Value Added Services Davidsson, P. 6.1 Executive Summary In a deregulated market the distribution utilities will compete with added value for the ***(multiagent*×*16, building*×*43)***
   hasID: 112
   hasAuthor: Davidsson, Paul
   hasDate: 10/1/1998 0:00:00
   hasProject: ISES Book, Chap.6
   hasTitle: Intelligent Buildings: Energy Saving and Value added Services
   hasPage: http://143.217.135.12/evaluation/knowledgebase/publications/books/ises/ISES6.htm
2. **Structured Engineering Process for Agent Communication Modelling**
   energy
   Structured Engineering Process for Agent Communication Modelling Hans Akkermans § Rune Gustavsson ‡ - Fredrik Ygge ¶ - § - AKMC and Free University Amsterdam Vuture.net ***(multiagent*×*40, building*×*4)***
3. **Structured Engineering Process for Agent Communication Modelling**
   energy
   Structured Engineering Process for Agent Communication Modelling Hans Akkermans § Rune Gustavsson ‡ - Fredrik Ygge ¶ - § - AKMC and Free University Amsterdam Vuture.net ***(multiagent*×*40, building*×*4)***
4. **Structured Engineering Process for Agent Communication Modelling**
   energy
   Structured Engineering Process for Agent Communication Modelling Hans Akkermans § Rune Gustavsson ‡ - Fredrik Ygge ¶ - § - AKMC and Free University Amsterdam Vuture.net ***(multiagent*×*40, building*×*4)***
5. **Structured Engineering Process for Agent Communication Modelling**
   energy
   Structured Engineering Process for Agent Communication Modelling Hans Akkermans § Rune Gustavsson ‡ - Fredrik Ygge ¶ - § - AKMC and Free University

Figure 11.2: The query interface of *Quiz*RDF

"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, *Quiz*RDF 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").

*Quiz*RDF is developed by the project partner BT, UK[8].

[8]BT, see `http://www.bt.com`

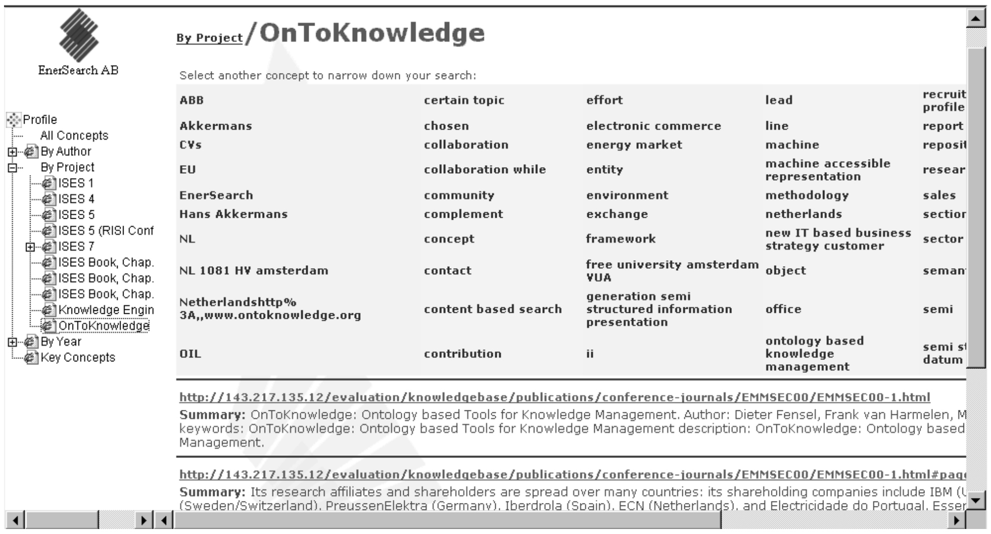
OntoShare: Community Support

OntoShare (Davies et al., 2002a; 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 11.20, taken from the BT case study).

OntoShare is like *Quiz*RDF developed by the project partner BT, UK.

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 does not only mean 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.



By Project/OnToKnowledge

Select another concept to narrow down your search:

ABB	certain topic	effort	lead	recruit profile
Akkermans	chosen	electronic commerce	line	report
CVs	collaboration	energy market	machine	reposit
EU	collaboration while	entity	machine accessible representation	resear
EnerSearch	community	environment	methodology	sales
Hans Akkermans	complement	exchange	netherlands	sectior
NL	concept	framework	new IT based business strategy customer	sector
NL 1081 HV amsterdam	contact	free university amsterdam VUA	object	seman
Netherlandshttp% 3A,,www.ontoknowledge.org	content based search	generation semi structured information presentation	office	semi
OIL	contribution	ii	ontology based knowledge management	semi st datum

http://143.217.135.12/evaluation/knowledgebase/publications/conference-journals/EMMSEC00/EMMSEC00-1.html
Summary: OnToKnowledge: Ontology based Tools for Knowledge Management. Author: Dieter Fensel, Frank van Harmelen, M
keywords: OnToKnowledge: Ontology based Tools for Knowledge Management description: OnToKnowledge: Ontology based
Management.

http://143.217.135.12/evaluation/knowledgebase/publications/conference-journals/EMMSEC00/EMMSEC00-1.html#page
Summary: Its research affiliates and shareholders are spread over many countries: its shareholding companies include IBM (U
(Sweden/Switzerland), PreussenElektra (Germany), Iberdrola (Spain), ECN (Netherlands), and Electricidade do Portugal. Esser

Figure 11.3: Provision of navigational structures with Spectacle

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.

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 11.3, 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.

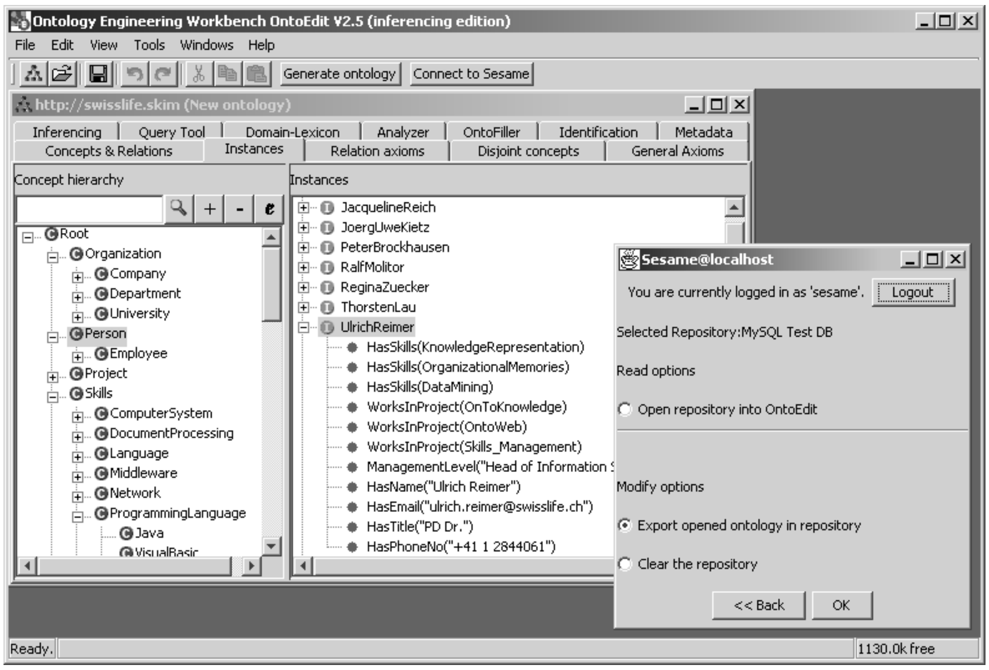
Spectacle is developed by the project partner AIdministrator, NL[9].

OntoEdit: Ontology Development

OntoEdit is presented in detail in Part III and the Appendices A, B and C. We here give a brief overview for readers that skipped those.

[9]AIdministrator, see `http://www.aidministrator.nl`

OntoEdit (Sure et al., 2002a; 2002b; Sure & Studer, 2001a) is a collaborative ontology engineering environment that is easily expandable through a flexible plugin framework (Handschuh, 2001). OntoEdit supports ontology engineers while inspecting, browsing, codifying and modifying ontologies in each step of the Knowledge Meta Process (*cf.* Chapter 4 or, *e.g.* (Staab et al., 2001)).


Ontology Engineering Workbench OntoEdit V2.5 (inferencing edition)
File Edit View Tools Windows Help
Generate ontology
Connect to Sesame
http://swisslife.skim (New ontology)
Inferencing | Query Tool | Domain-Lexicon | Analyzer | OntoFiller | Identification | Metadata
Concepts & Relations | Instances | Relation axioms | Disjoint concepts | General Axioms
Concept hierarchy
Instances
Sesame@localhost
Logout
<< Back
OK
Ready.
1130.0k free

Concept hierarchy:

```
Root
  Organization
    Company
    Department
    University
  Person
    Employee
  Project
  Skills
    ComputerSystem
    DocumentProcessing
    Language
    Middleware
    Network
    ProgrammingLanguage
      Java
      VisualBasic
```

Instances:

```
JacquelineReich
JoergUweKietz
PeterBrockhausen
RalfMolitor
ReginaZuecker
ThorstenLau
UlrichReimer
  HasSkills(KnowledgeRepresentation)
  HasSkills(OrganizationalMemories)
  HasSkills(DataMining)
  WorksInProject(OnToKnowledge)
  WorksInProject(OntoWeb)
  WorksInProject(Skills_Management)
  ManagementLevel("Head of Information S
  HasName("Ulrich Reimer")
  HasEmail("ulrich.reimer@swisslife.ch")
  HasTitle("PD Dr.")
  HasPhoneNo("+41 1 2844061")
```

You are currently logged in as 'sesame'.

Selected Repository:MySQL Test DB

Read options

( ) Open repository into OntoEdit

Modify options

(•) Export opened ontology in repository

( ) Clear the repository

Figure 11.4: Ontology development with OntoEdit

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), Sesame Plugin, OntoFiller and, last but not least, OntoClean Plugin (Sure et al., 2002b). They are explained further in Chapter 4.

Figure 11.4 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[10] (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:

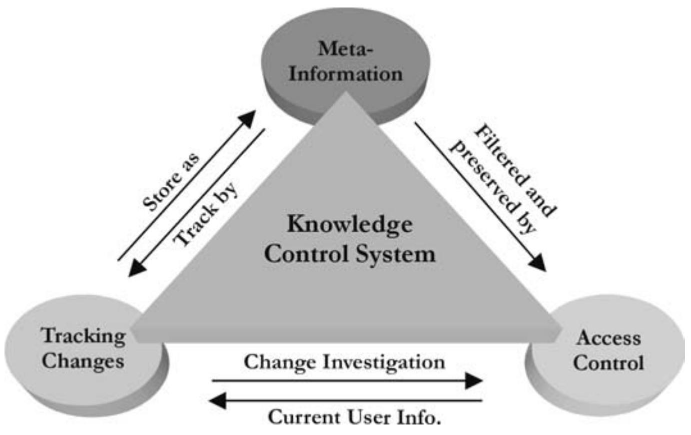


Figure 11.5: Features of the Knowledge Control System

- Change management for ontologies to allow for 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;

[10]More information for OMM and BOR including an online demo can be found at (OntoText, 2003c; 2003a; 2003b).

- 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)[11]) 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. OMM was 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 11.5), additional support for multi-protocol access (*e.g.* HTTP, RMI, SOAP) and reasoning services.

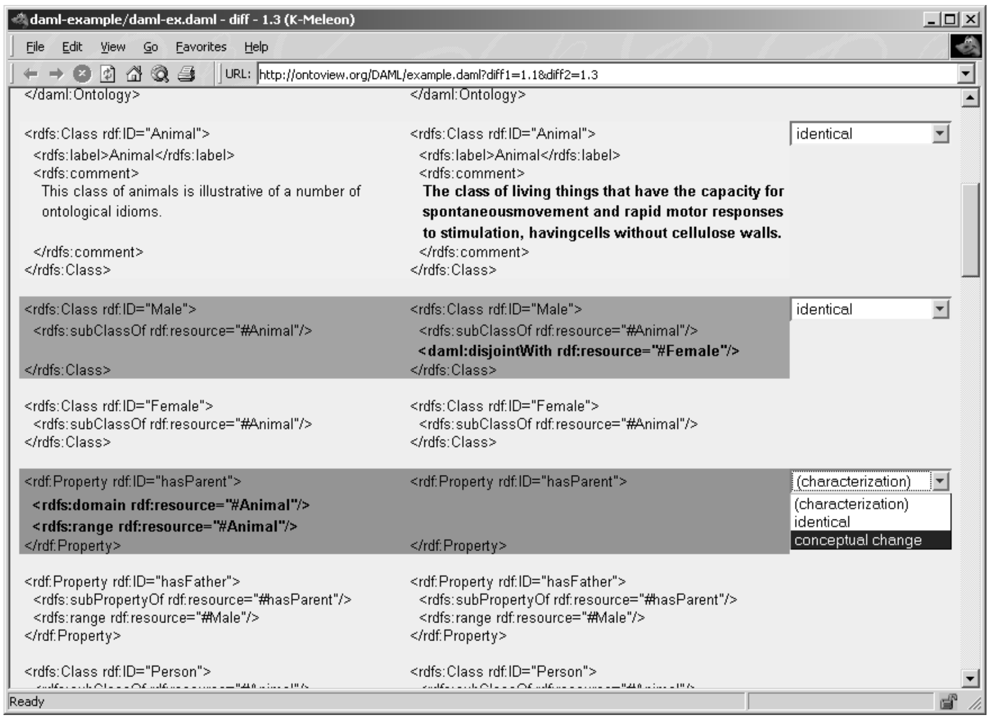
An example for a reasoning service is BOR (Simov & Jordanov, 2002) – a reasoner that is currently being developed and complies with the DAML+OIL model-theoretic semantics. It is a modular component that can be plugged in to extend the query facilities already provided *e.g.* by Sesame. It addresses most 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.

OMM is developed by the project partner Sirma AI / OntoText Lab., BG[12].

OntoView: Change Management for Ontologies

OntoView (Klein et al., 2002b; 2002a) is a change management tool for ontologies and is implemented as part of the Ontology Middleware Module (it is not separately shown in Figure 11.1). 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.

[11]http://www.cvshome.org/
[12]Sirma AI / OntoText Lab., see `http://www.sirma.bg`


daml-example/daml-ex.daml - diff - 1.3 (K-Meleon)
File Edit View Go Favorites Help
URL: http://ontoview.org/DAML/example.daml?diff1=1.1&diff2=1.3
Ready

```
</daml:Ontology>                                           </daml:Ontology>

<rdfs:Class rdf:ID="Animal">                               <rdfs:Class rdf:ID="Animal">                          identical
  <rdfs:label>Animal</rdfs:label>                            <rdfs:label>Animal</rdfs:label>
  <rdfs:comment>                                             <rdfs:comment>
   This class of animals is illustrative of a number of       The class of living things that have the capacity for
   ontological idioms.                                        spontaneousmovement and rapid motor responses
                                                              to stimulation, havingcells without cellulose walls.
  </rdfs:comment>                                            </rdfs:comment>
</rdfs:Class>                                              </rdfs:Class>

<rdfs:Class rdf:ID="Male">                                 <rdfs:Class rdf:ID="Male">                            identical
  <rdfs:subClassOf rdf:resource="#Animal"/>                  <rdfs:subClassOf rdf:resource="#Animal"/>
                                                             <daml:disjointWith rdf:resource="#Female"/>
</rdfs:Class>                                              </rdfs:Class>

<rdfs:Class rdf:ID="Female">                               <rdfs:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>                  <rdfs:subClassOf rdf:resource="#Animal"/>
</rdfs:Class>                                              </rdfs:Class>

<rdf:Property rdf:ID="hasParent">                          <rdf:Property rdf:ID="hasParent">                     (characterization)
  <rdfs:domain rdf:resource="#Animal"/>                                                                            (characterization)
  <rdfs:range rdf:resource="#Animal"/>                                                                             identical
</rdf:Property>                                            </rdf:Property>                                         conceptual change

<rdf:Property rdf:ID="hasFather">                          <rdf:Property rdf:ID="hasFather">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>            <rdfs:subPropertyOf rdf:resource="#hasParent"/>
  <rdfs:range rdf:resource="#Male"/>                         <rdfs:range rdf:resource="#Male"/>
</rdf:Property>                                            </rdf:Property>

<rdfs:Class rdf:ID="Person">                               <rdfs:Class rdf:ID="Person">
```

Figure 11.6: The result of a comparison of two ontologies with OntoView

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 11.6.

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 11.6 (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 sub-property of "$\text{hasParent}_{2.0}$".

Another function is the possibility to analyze 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., 2000b). 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.

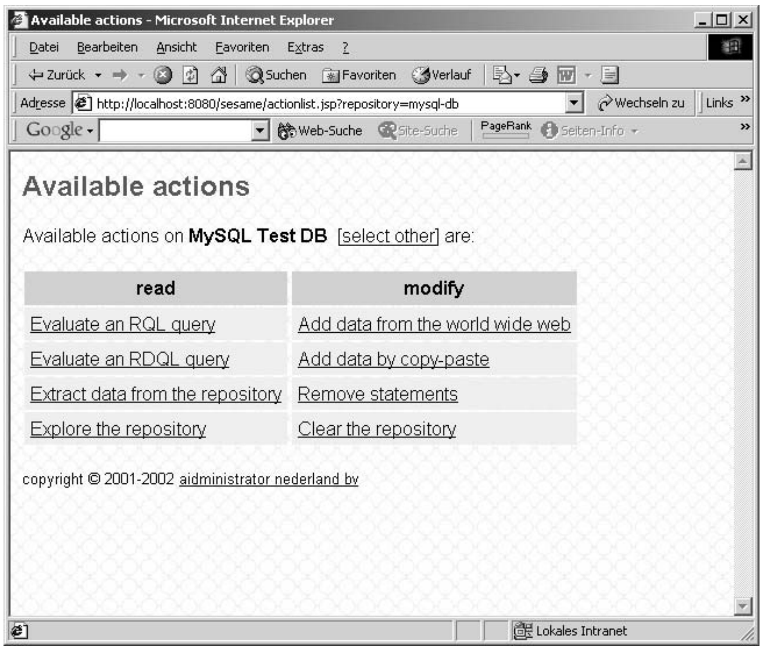
OntoView is developed by the project partners Free University of Amsterdam (VUB), NL[13], and Sirma AI / OntoText Lab., BG[14].

Sesame: Repository for Ontologies and Data

Sesame[15] (Broekstra et al., 2002; Broekstra & Kampman, 2001) is a system that allows persistent storage of RDF data and schema information and subsequent online

[13]VUB, see `http://www.cs.vu.nl`
[14]Sirma AI / OntoText Lab., see `http://www.sirma.bg`
[15]More information including an online demo can be found at (AIdministrator, 2003b), the Source Forge project website can be found at (AIdministrator, 2003a)

querying of that information. Sesame has been designed with scalability, portability and extensibility in mind.


Available actions - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Verlauf
Adresse http://localhost:8080/sesame/actionlist.jsp?repository=mysql-db
Wechseln zu Links
Google Web-Suche Site-Suche PageRank Seiten-Info
Lokales Intranet

Available actions

Available actions on **MySQL Test DB** [select other] are:

read	modify
Evaluate an RQL query	Add data from the world wide web
Evaluate an RDQL query	Add data by copy-paste
Extract data from the repository	Remove statements
Explore the repository	Clear the repository

copyright © 2001-2002 aidministrator nederland bv

Figure 11.7: 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 Source under the GNU Lesser General Public License (LGPL).

Figure 11.7 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.

Sesame is like Spectacle developed by the project partner AIdministrator, NL.

CORPORUM: Information Extraction

The CORPORUM toolset (Engels & Bremdal, 2001a; 2000; 2001b; 2002) consists of two parts, *viz.* OntoExtract (*cf.* Figure 11.8 showing some options of OntoExtract for automatic ontology generation) and OntoWrapper (*cf.* Figure 11.9 showing the generation of templates for wrapping information sources). Hence, it has two related, though different, tasks: interpretation of natural language texts and extraction of specific information from free text.

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.*).

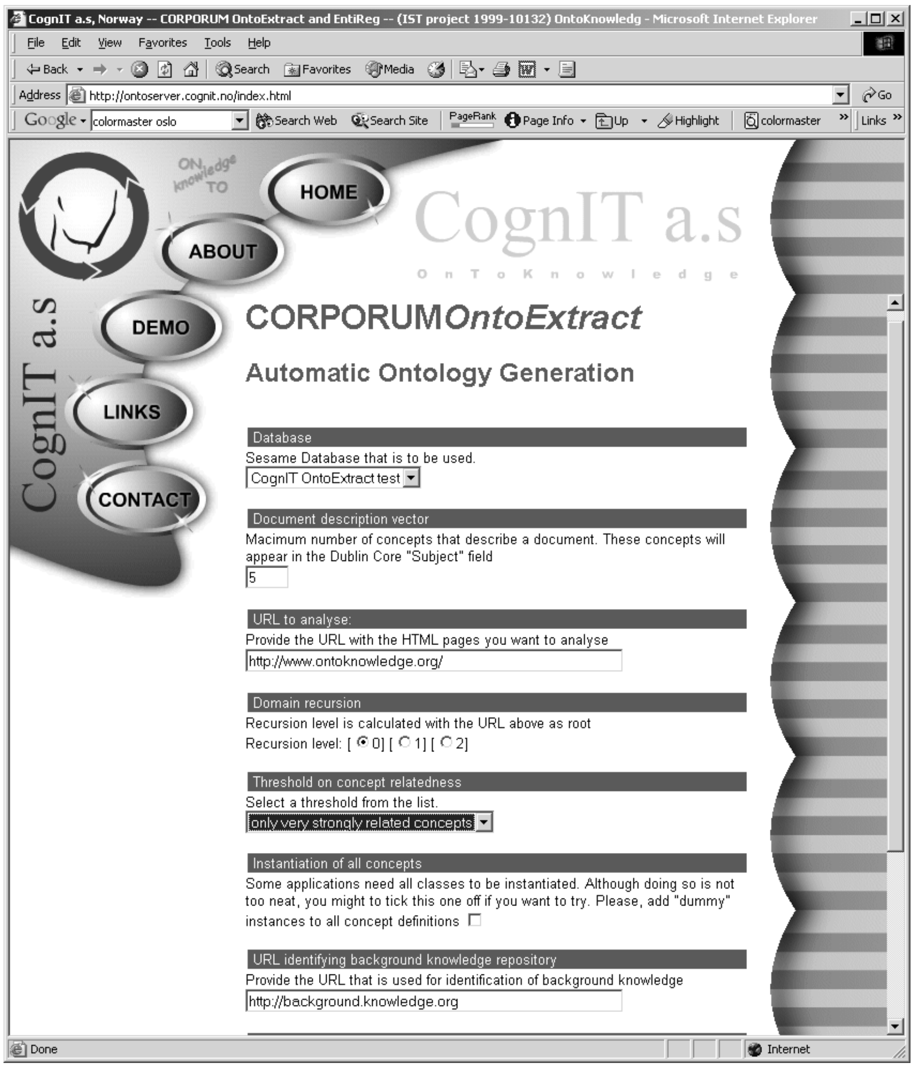
The CORPORUM system outputs a variety of (symbolic) knowledge representations, including semantic (network) structures and visualizations thereof, light-weight ontologies, text summaries, automatically generated thesauri (related words/concepts), etc. Thus, extracted information is represented in RDF(S), augmented with Dublin Core meta data (*cf.* (DC, 2003)) wherever possible, and submitted to the Sesame data repository mentioned previously.

CORPROUM is developed by the project partner CognIT, NO[16].

11.2.2 OIL: Ontology Inference Layer for the Semantic 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

[16]CognIT, see `http://www.cognit.no`


CognIT a.s, Norway -- CORPORUM OntoExtract and EntiReg -- (IST project 1999-10132) OntoKnowledg - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://ontoserver.cognit.no/index.html Go
Google colormaster oslo Search Web Search Site PageRank Page Info Up Highlight colormaster Links
ON knowledge TO
HOME
ABOUT
DEMO
LINKS
CONTACT
CognIT a.s
CognIT a.s
O n T o K n o w l e d g e
Done
Internet

CORPORUM*OntoExtract*

Automatic Ontology Generation

Database
Sesame Database that is to be used.
CognIT OntoExtract test

Document description vector
Macimum number of concepts that describe a document. These concepts will appear in the Dublin Core "Subject" field
5

URL to analyse:
Provide the URL with the HTML pages you want to analyse
http://www.ontoknowledge.org/

Domain recursion
Recursion level is calculated with the URL above as root
Recursion level: [[✓ 0]] [1] [2]

Threshold on concept relatedness
Select a threshold from the list.
only very strongly related concepts

Instantiation of all concepts
Some applications need all classes to be instantiated. Although doing so is not too neat, you might to tick this one off if you want to try. Please, add "dummy" instances to all concept definitions ☐

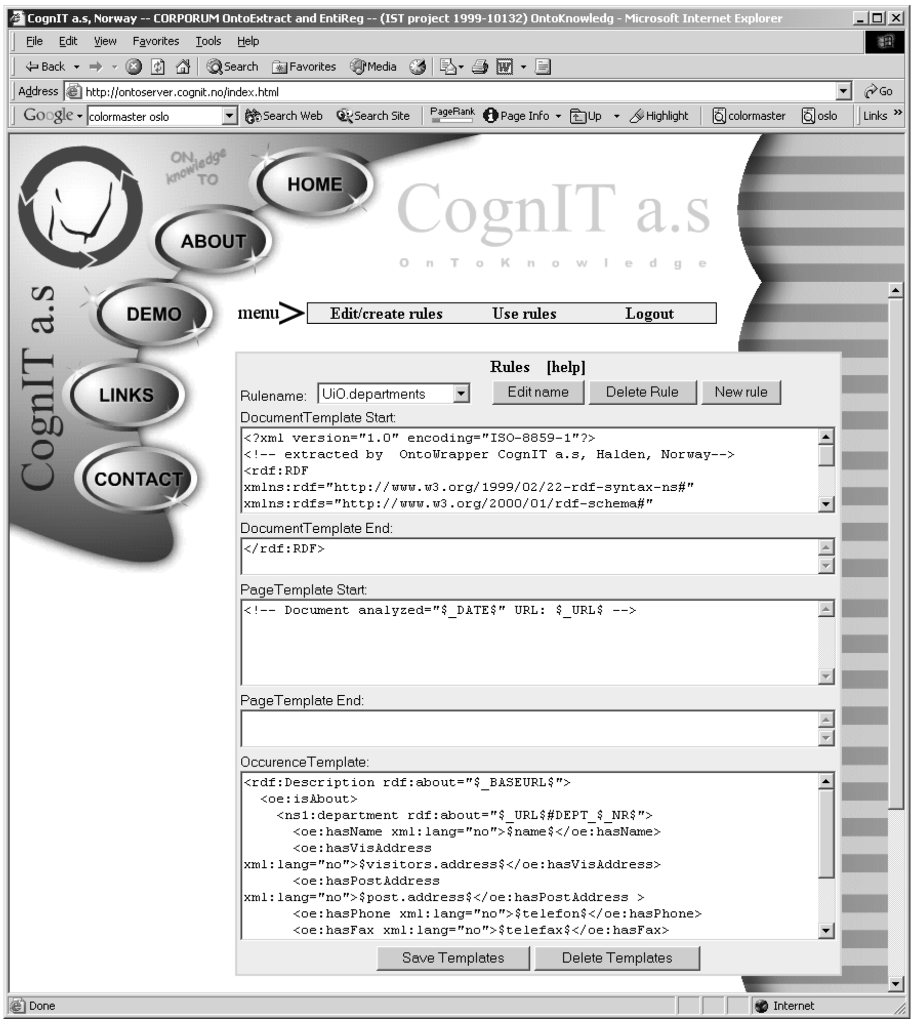
URL identifying background knowledge repository
Provide the URL that is used for identification of background knowledge
http://background.knowledge.org

Figure 11.8: OntoExtract: Automatic ontology generation

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 (*Quiz*RDF) 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



Rulename: UiO.departments

DocumentTemplate Start:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- extracted by  OntoWrapper CognIT a.s, Halden, Norway-->
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

DocumentTemplate End:

```
</rdf:RDF>
```

PageTemplate Start:

```
<!-- Document analyzed="$_DATE$" URL: $_URL$ -->
```

PageTemplate End:

OccurenceTemplate:

```
<rdf:Description rdf:about="$_BASEURL$">
  <oe:isAbout>
    <ns1:department rdf:about="$_URL$#DEPT_$_NR$">
      <oe:hasName xml:lang="no">$name$</oe:hasName>
      <oe:hasVisAddress
xml:lang="no">$visitors.address$</oe:hasVisAddress>
      <oe:hasPostAddress
xml:lang="no">$post.address$</oe:hasPostAddress >
      <oe:hasPhone xml:lang="no">$telefon$</oe:hasPhone>
      <oe:hasFax xml:lang="no">$telefax$</oe:hasFax>
```

Figure 11.9: OntoWrapper: Information extraction

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 & Gómez-Pérez, 2000) for a recent overview), it is particularly the 3rd requirement that motivated the OTK consortium to design a new language (sometimes called "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., 2000c; 1999; 2000b; 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 RDF(S) based serializations (as well as a formally specified "human readable" form, which we will use here[17]). 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.
- 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.

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

- 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 shown in Sections 11.4, 11.5 and 11.6, 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 sub-language 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.

The standard language is called “Standard OIL”. When extended with the ability to assert that individuals and tuples are, respectively, instances of classes and slots, it

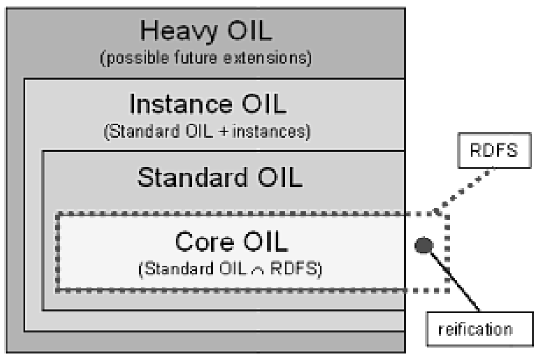


Figure 11.10: The layered language model of OIL

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 11.10.

Figure 11.11 illustrates an OIL ontology (using the human readable serialization), developed in a skills management case study by Swiss Life (*cf.* Section 11.4).

```
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 11.11: OIL illustration

The following points are noteworthy:

- `Skills` are restricted to being of a single level trough a cardinality constraint (*i.e.* a person can only have one particular skill level in any given skill),
- `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[18]. In November 2001, the W3C started a Working Group (WG) for defining a Web Ontology language[19]. This WG was 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 the 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).

11.3 Configuration of Tools in the Case Studies

Similar to LEGO[20] pieces the OTK tools can be plugged together in various ways to meet specific requirements. Each OTK case study has a different configuration that is briefly shown in the following subsections[21].

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

[19]*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

[20]The LEGO Group, see `http://www.lego.com/`

[21]It is noteworthy that BOR could not be explored in any case study due to the fact that the developing partner OntoText was introduced to the project at a late stage. The tool was designed and implemented in an impressively short period of time, but was only finished in the final phase of On-To-Knowledge.

By introducing the tools used in each case study, we already present the part on selection of tools of each feasibility study (*cf.* Sections 11.4, 11.5 and 11.6). By integrating the feasibility study parts into a separate section we aim at giving a general overview of the coverage of tools in the case studies.

11.3.1 Tools @ Swiss Life

The case study at Swiss Life was the first one to start, unfortunately it has been 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 (*cf.* Figure 11.12, the dark grey shaded tools were planned to be used).

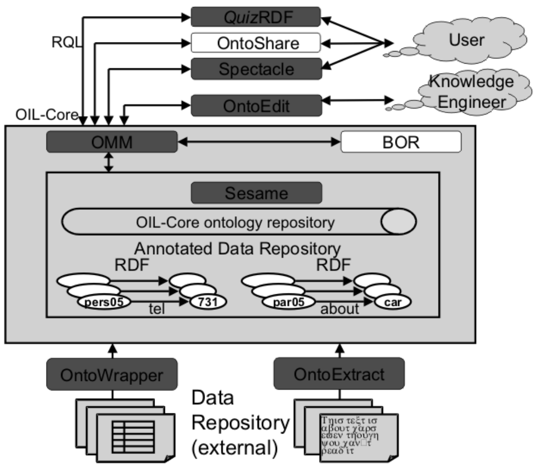


Figure 11.12: Covering of tools @ Swiss Life case study

The case study started right at the beginning of the project and was the first test ground for the early versions of the tools. On the one hand, the case study was a major kick-off that leaded in the end to deploying an up-and-running and integrated tool suite. On the other hand, since the tools were obviously not fully functional from the very beginning, the first prototype of the case study relied only to some extent on the tools that were envisioned for the final application (as shown in Figure 11.12).

OntoEdit was used to model the skills domain. The plugins OntoKick, Mind2Onto, OntoFiller and Sesame Plugin (*cf.* Sections 7.1, 7.2, 8.1 and 10.1) were actually initiated by requirements coming out of this first case study and they were partially used in it. Sesame served as central repository.

OntoExtract and OntoWrapper were envisioned to extract skill profiles out of existing documents, but they could only be tested. Same holds for *Quiz*RDF and Spectacle, they were tested as query and navigation interfaces on top of Sesame. OntoShare and the Ontology Middleware Module were developed rather late in the project. Similar to the EnerSearch case study, pre- and post-trials should help to evaluate the technology and the ontology itself.

11.3.2 Tools @ BT

The BT case study is centered around OntoShare (*cf.* Figure 11.13, the dark grey shaded tools are used in this case study).

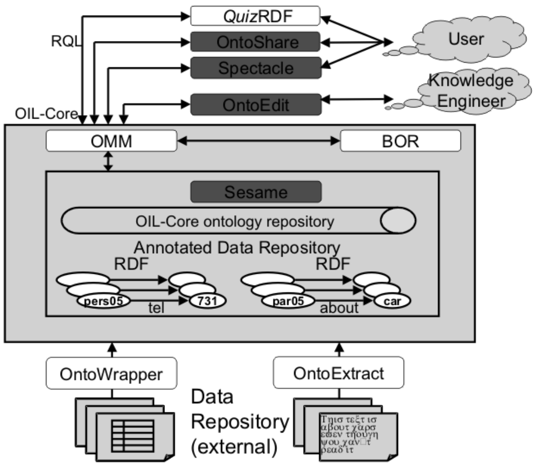


Figure 11.13: Covering of tools @ BT case study

OntoEdit was used to develop the underlying ontology, Sesame served as central storage and query engine (*e.g.* the user profiles of OntoShare are derived out of RQL queries (*cf.* Section 11.2.1)). Though not included into the evaluation (*cf.* Section 11.5), we used the information presentation layer of Spectacle as a web-based access to OntoShare and *Quiz*RDF as an enhanced query facility.

11.3.3 Tools @ EnerSearch

Main parts of the EnerSearch case study consisted of a user-focused and technology-focused evaluation of the OTK tools (*cf.* Figure 11.13, the dark grey shaded tools are used in this case study).

Firstly, in a field experiment *Quiz*RDF and Spectacle were compared against the EnerSEARCHer, a traditional keyword based search engine.

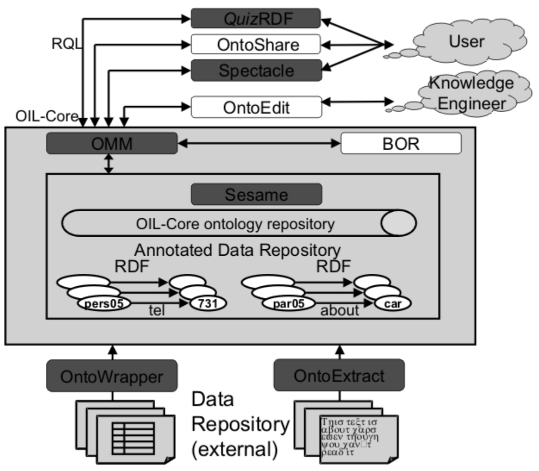


Figure 11.14: Covering of tools @ EnerSearch case study

Secondly, Sesame and the Ontology Middleware Module were tested on scalability and interoperability issues. OntoExtract and OntoWrapper, both part of the CORPORUM tool suite, provided the necessary support for generating automatically ontologies for the user-focussed scenario.

11.4 Skills Management @ Swiss Life

11.4.1 Introduction

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 (*cf.* (Lau & Sure, 2002)[22]) explores different parts of the intranet with a main focus on the introduction of an advanced skills management application.

We now describe how we instantiated in close cooperation with our project partner Swiss Life the On-To-Knowledge Methodology (presented in Part II) in the case study.

[22]Further information on the Swiss Life case study can be found in the On-To-Knowledge deliverables (Novotny & Lau, 2000; 2001; Novotny et al., 2001)

11.4.2 Feasibility Study

Problems & Opportunities

"Skills Management" (Edvinson & Malone, 1997; Stader & Macintosh, 1999) makes skills of employees explicit. There exist only few publications that describe sophisticated skills management applications in detail (*cf.*, *e.g.*, (Liao et al., 1999; Becerra-Fernandez, 2000; Sure et al., 2000; Maedche, 2002b); (Dittmann et al., 2003) addresses the social aspects implied by such sophisticated applications), but many companies internally build up their own skills management applications, typically on top of database systems (*cf.*, *e.g.*, (Elbert, 2001)).

The following typical objectives of skills management were taken as a starting point to develop own application scenarios. The objectives are grouped by their respective organizational level.

- Employees:
  - Help in finding suitable knowledge for specific problems.
  - Support in developing own skills in a most focused way. This includes support for planning ones own career.
- Management:
  - Search of suitable employees for projects and positions (staffing).
  - Overview of current skills and development needs/potential of employees.
- Company:
  - Find the right employee for the right position to enhance the productivity of the company.
  - HR development measures can be deduced by comparing the strategy with employees' skills and thereby detecting missing skills.
  - Measure and improve the intellectual capital of the company.

We followed the principle "Start small, but think big!" and created a roadmap (*cf.* Figure 11.15) that increases stepwise the required level of effort and the level of complexity. As shown in the following enumeration, we start with support for employees at their desktop and end with support for the decision making process at the strategic level of the company.

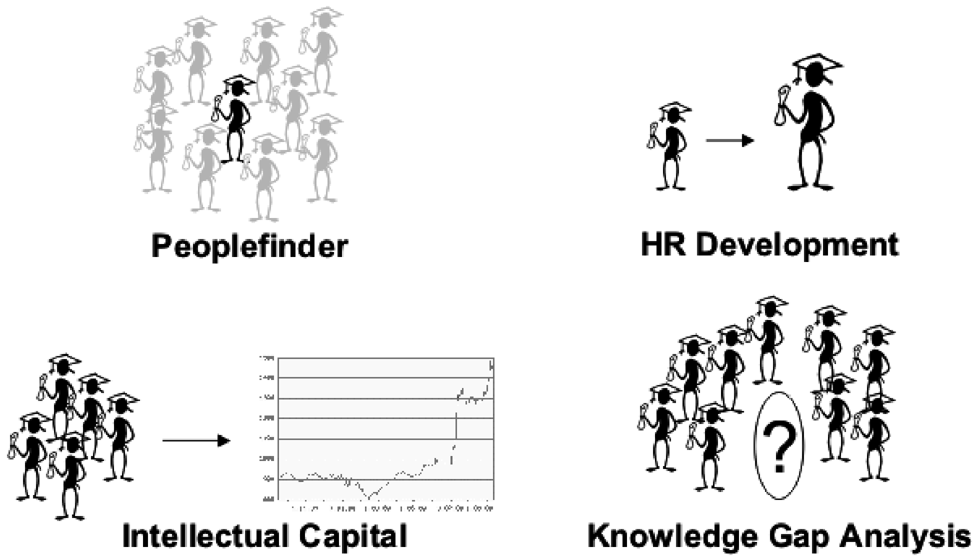


Figure 11.15: Roadmap for Skills Management @ Swiss Life

1. **Peoplefinder:** With the switch from line organisation to project organisation and shorter project running times, assembling staff for a project team is increasingly becoming a management task. This requires an excellent overview of the skills of the available staff. Making skills explicit allows for an advanced expert search within the intranet.

2. **Human Resource (HR) Development:** Similar to skill profiles there exist also job profiles. Employees might explore their future career path by matching their current skill profile *vs.* job profiles. An extension is the subsequent offering of internal learning modules, that help employees in reaching required levels of expertise that are required for certain jobs.

3. **Knowledge Gap Analysis:** Having available all skills of employees, a company is able to find out knowledge gaps. For instance if a new project is planned one might find out whether all required skills are available in the company[23]. Thus, a company can detect in which areas they might need to improve the skills of their employees.

4. **Intellectual Capital:** Measuring and improving the intellectual capital might help to improve the shareholder value of a company. Explicit skill profiles as

[23]Of course an equally important aspect that is not covered by this item is to find out whether the person having the skills is available for the project.

an integral part of corporate decision support systems enables an up-to-date measuring of skills.

Within Swiss Life there already existed several applications, typically databases, that contained valuable information about employees, *e.g.* their contact information, their functions and roles and even (for some of them) skill profiles in a legacy skills database. The integration of these different knowledge sources offered large potential for the case study:

- Existing knowledge can be re-used and does not have to be captured again.
- The maintenance of each source is already organized and therefore the maintenance task of the skills management application is significantly reduced.

These benefits made a "Quick Win" solution[24] feasible as a starting point for the case study.

To ensure that all integrated knowledge sources were used in the same way, ontologies were 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 same concepts, *e.g.* "WinWord" *vs.* "MS Word".

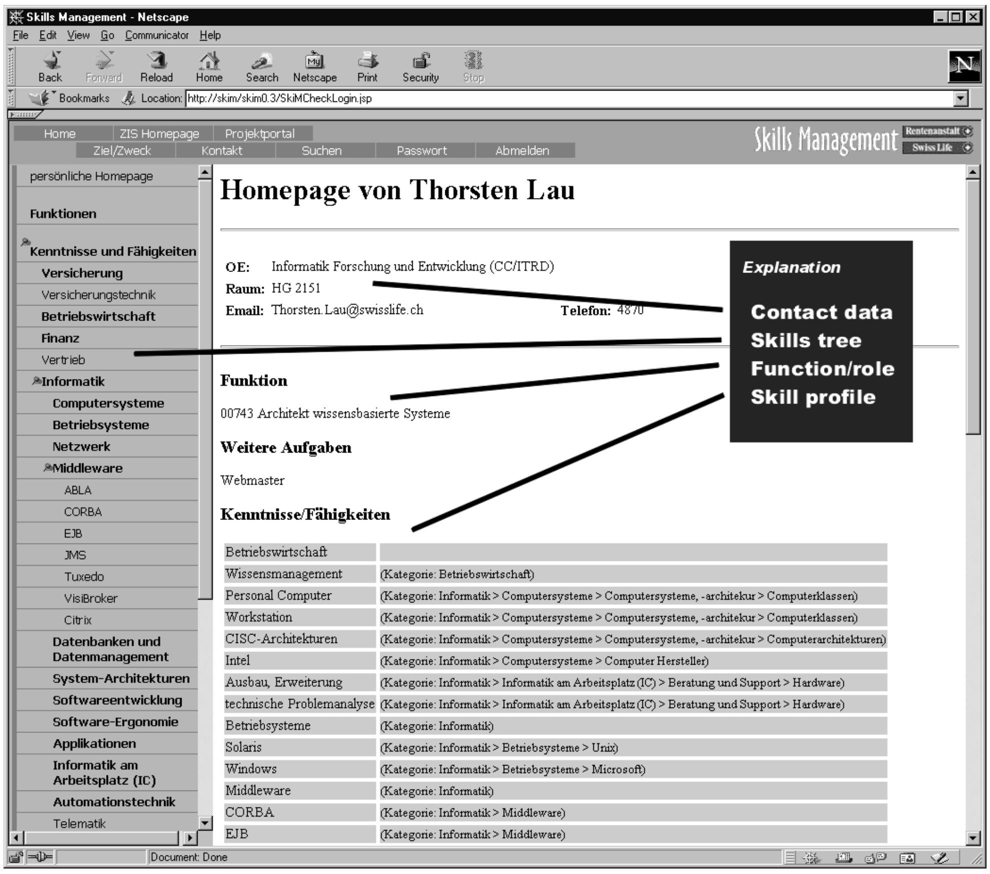
To tackle these problems, the development of OntoFiller, *cf.* Section 8.1, and of the Domain Lexicon, *cf.* Section 7.3, were initiated.

Focus of KM Application

This case study serves as a starting point for introducing skills management. The **peoplefinder** scenario was therefore the main focus of the resulting KM application.

To illustrate the application we already present a screenshot of the developed prototype in Figure 11.16. 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:

[24]It is often argued that the introduction of new knowledge management solutions should start with "Quick Wins" that show rather quickly and without much effort the potential value of KM.



Homepage von Thorsten Lau

OE: Informatik Forschung und Entwicklung (CC/ITRD)
Raum: HG 2151
Email: Thorsten.Lau@swisslife.ch **Telefon:** 4870

Funktion

00743 Architekt wissensbasierte Systeme

Weitere Aufgaben

Webmaster

Kenntnisse/Fähigkeiten

Betriebswirtschaft	
Wissensmanagement	(Kategorie: Betriebswirtschaft)
Personal Computer	(Kategorie: Informatik > Computersysteme > Computersysteme, -architekur > Computerklassen)
Workstation	(Kategorie: Informatik > Computersysteme > Computersysteme, -architekur > Computerklassen)
CISC-Architekturen	(Kategorie: Informatik > Computersysteme > Computersysteme, -architekur > Computerarchitekturen)
Intel	(Kategorie: Informatik > Computersysteme > Computer Hersteller)
Ausbau, Erweiterung	(Kategorie: Informatik > Informatik am Arbeitsplatz (IC) > Beratung und Support > Hardware)
technische Problemanalyse	(Kategorie: Informatik > Informatik am Arbeitsplatz (IC) > Beratung und Support > Hardware)
Betriebsysteme	(Kategorie: Informatik)
Solaris	(Kategorie: Informatik > Betriebsysteme > Unix)
Windows	(Kategorie: Informatik > Betriebsysteme > Microsoft)
Middleware	(Kategorie: Informatik)
CORBA	(Kategorie: Informatik > Middleware)
EJB	(Kategorie: Informatik > Middleware)

Figure 11.16: Skills management prototype

- 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
- 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..." (not shown in Figure 11.16, the section is below the skill profile).

Last, but not least, the complete skills tree is available for navigation and browsing on the left side.

People

The prototype should reflect the skills of employees of different departments. Obviously, the management of those departments were core stakeholders in the case study.

A major effort was the development of an appropriate ontology for the skills management application. Therefore not only management support for the application but also the help of domain experts from departments was needed to capture relevant aspects of their domain. Three departments took part in the case study: Human Resource, Private Insurance and Information Technology. From each department a domain expert was named by the management as additional stakeholder who supported the ontology development.

11.4.3 Kickoff

Capture Requirements Specification

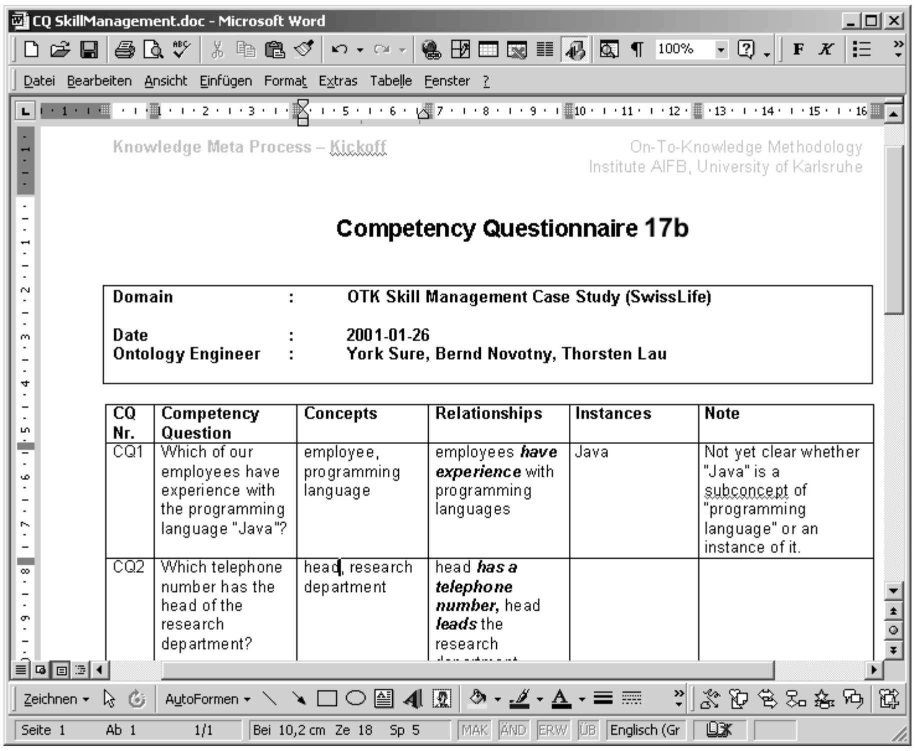
The ontology for the envisaged skills management application needed to reflect two main requirements:

- Integration of existing sources, therefore the ontology needs to reflect the relevant conceptual structures of the integrated sources.
- Representation of skills relevant for the three participating departments and their domains.

We started by capturing competency questions that cover the "competency" an ontology should have to answer certain questions. Initially that was simply done in a Word document as shown in Figure 11.17. An example for a competency question is: "Which of our employees have experience with the programming language 'Java'?".

In the formalization step, one might derive the concepts Employee and Programming Language, the relationship HAVE EXPERIENCE with the domain Employee and the range Programming Language and the instance `Java` as a particular Programming Language out of the competency question. However, the actual modelling of particular concepts, relationships and instances is done during the formalization into the target ontology.

The extraction of concepts and relations out of the competency questions does not exploit the competency questions to their full potential. During later stages of the ontology development lifecycle the link back from concepts (or relations) to the competency questions gives valuable information why a certain concept or relationship is modelled, *i.e.* it provides more context. Especially when the ontology evolves over time


CQ SkillManagement.doc - Microsoft Word
Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ?
Zeichnen AutoFormen
Seite 1 Ab 1 1/1 Bei 10,2 cm Ze 18 Sp 5 MAK ÄND ERW ÜB Englisch (Gr

Knowledge Meta Process – Kickoff

On-To-Knowledge Methodology
Institute AIFB, University of Karlsruhe

Competency Questionnaire 17b

Domain	:	**OTK Skill Management Case Study (SwissLife)**
Date	:	**2001-01-26**
Ontology Engineer	:	**York Sure, Bernd Novotny, Thorsten Lau**

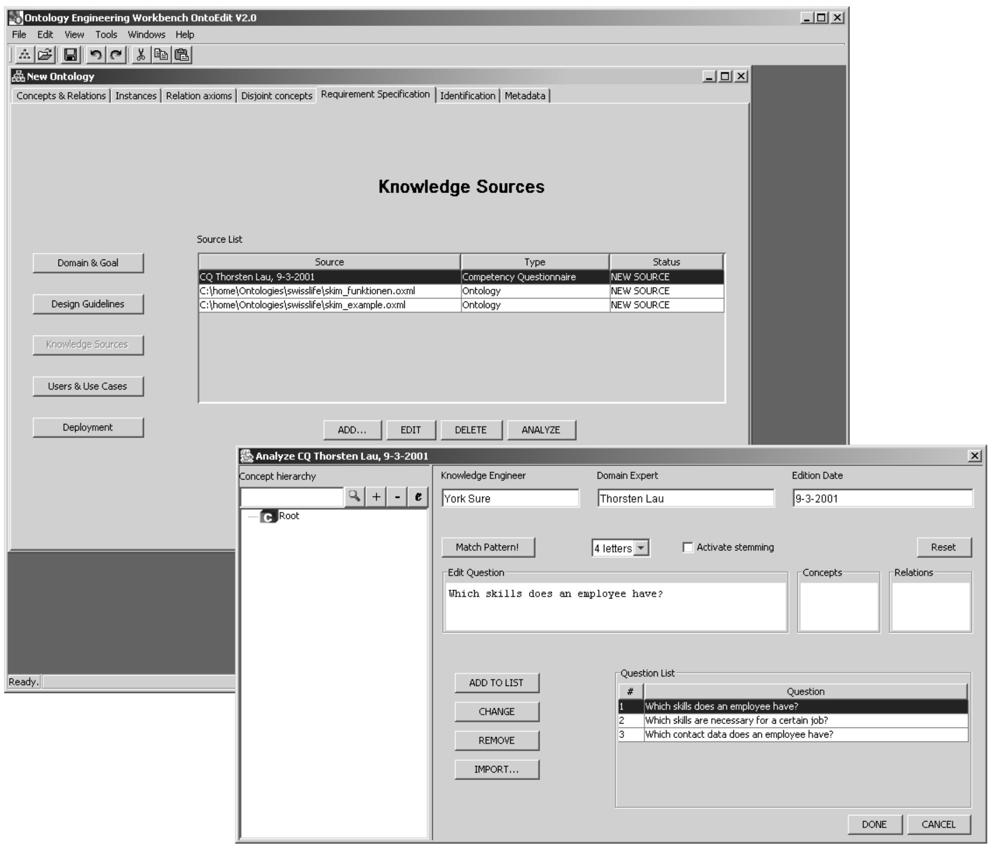
CQ Nr.	**Competency Question**	**Concepts**	**Relationships**	**Instances**	**Note**
CQ1	Which of our employees have experience with the programming language "Java"?	employee, programming language	employees ***have experience*** with programming languages	Java	Not yet clear whether "Java" is a subconcept of "programming language" or an instance of it.
CQ2	Which telephone number has the head of the research department?	head, research department	head ***has a telephone number***, head ***leads*** the research		

Figure 11.17: Competency questionnaire @ Swiss Life

and people are not so familiar any more with their initial design decisions, this linkages gives some kind of explanation. Therefore the OntoKick plugin was developed (*cf.* Section 7.1), that captures the requirements for ontologies including the competency questions within OntoEdit, keeps track of the extraction of concepts and relations and stores the requirements and links along with the ontology. An example of the competency questions in OntoKick is shown in Figure 11.18.

First experiments with extracting an ontology semi-automatically by using typical information extraction tools did not satisfy the needs for a clearly structured and easily understandable model of the skills. The domain experts and potential users felt very uncomfortable with the extracted structures and chose rather to build the ontology "manually".

It was agreed upon the modelling experts and the domain experts to start with a rather "lightweight" ontology that consists mainly of a tree-like structure of skills and to extend it afterwards with cross-taxonomical relationships.


Ontology Engineering Workbench OntoEdit V2.0
File Edit View Tools Windows Help
New Ontology
Concepts & Relations | Instances | Relation axioms | Disjoint concepts | Requirement Specification | Identification | Metadata
Domain & Goal
Design Guidelines
Knowledge Sources
Users & Use Cases
Deployment
ADD... EDIT DELETE ANALYZE
Ready.
Analyze CQ Thorsten Lau, 9-3-2001
Concept hierarchy
Root
Match Pattern!
Reset
ADD TO LIST
CHANGE
REMOVE
IMPORT...
DONE
CANCEL

Knowledge Sources

Source List

Source	Type	Status
CQ Thorsten Lau, 9-3-2001	Competency Questionnaire	NEW SOURCE
C:\home\Ontologies\swisslife\skim_funktionen.oxml	Ontology	NEW SOURCE
C:\home\Ontologies\swisslife\skim_example.oxml	Ontology	NEW SOURCE

Knowledge Engineer: York Sure
Domain Expert: Thorsten Lau
Edition Date: 9-3-2001

4 letters
Activate stemming

Edit Question: Which skills does an employee have?

Concepts

Relations

Question List

#	Question
1	Which skills does an employee have?
2	Which skills are necessary for a certain job?
3	Which contact data does an employee have?

Figure 11.18: The skills management competency questionnaire in OntoKick

Create Semi-Formal Ontology Description

During the kickoff phase two workshops with three domain experts[25] are held. The first one introduces the domain experts to the ideas of ontologies. Additional potential knowledge sources are identified by the domain experts, that are exhaustively used for the development of the ontologies, *e.g.* a book of the Swiss Association of Data Processing ("Schweizerischer Verband fuer Datenverarbeitung") describing professions in the computing area in a systematic way similar to an ontology.

To develop the semi-formal description of the ontology we used a commercially available mind mapping tool. It is typically used for brainstorming sessions and provides simple facilities for modelling hierarchies very quickly and intuitively. The early modelling stages for ontologies contain elements from such brainstorming sessions, therefore

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

such tool support is very helpful. This triggered the development of the Mind2Onto framework, *cf.* Section 7.2, to bridge the gap between semi-formal ontology description and target ontology.

11.4.4 Refinement

Refine Semi-Formal Description of Ontology

During the refinement phase we integrated and formalized the semi-formal ontology descriptions into a single coherent skills ontology. The domain experts initially developed three different skill mind maps™, for each department a separate one. During the refinement phase it was necessary to combine the three into a single coherent one.

An important first step was the elimination of inconsistencies and of 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 uses 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. As known from personal contacts to other companies (*e.g.* Credit Suisse, ABB and IBM), such an approach proved to produce highly reliable information.

Figure 11.19 gives an impression of the size the mind map™ reached rather quickly. Within two weeks the domain experts modelled around 700 skills in a tree-like structure.

Formalization into Target 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 HAS SUBTOPIC relationship. Common to both is the transitivity, but they differ in their inheritance capabilities. The HAS SUBTOPIC relationship does not imply the inheritance of relations from the superconcept to the subconcept of the relationship, the IS A relationship does.

However, for our first prototype this distinction made no difference due to missing cross-taxonomical relationships, there simply was nothing to inherit. But, according to (Guarino & Welty, 2002), subsumption provided by IS A taxonomies is often misused

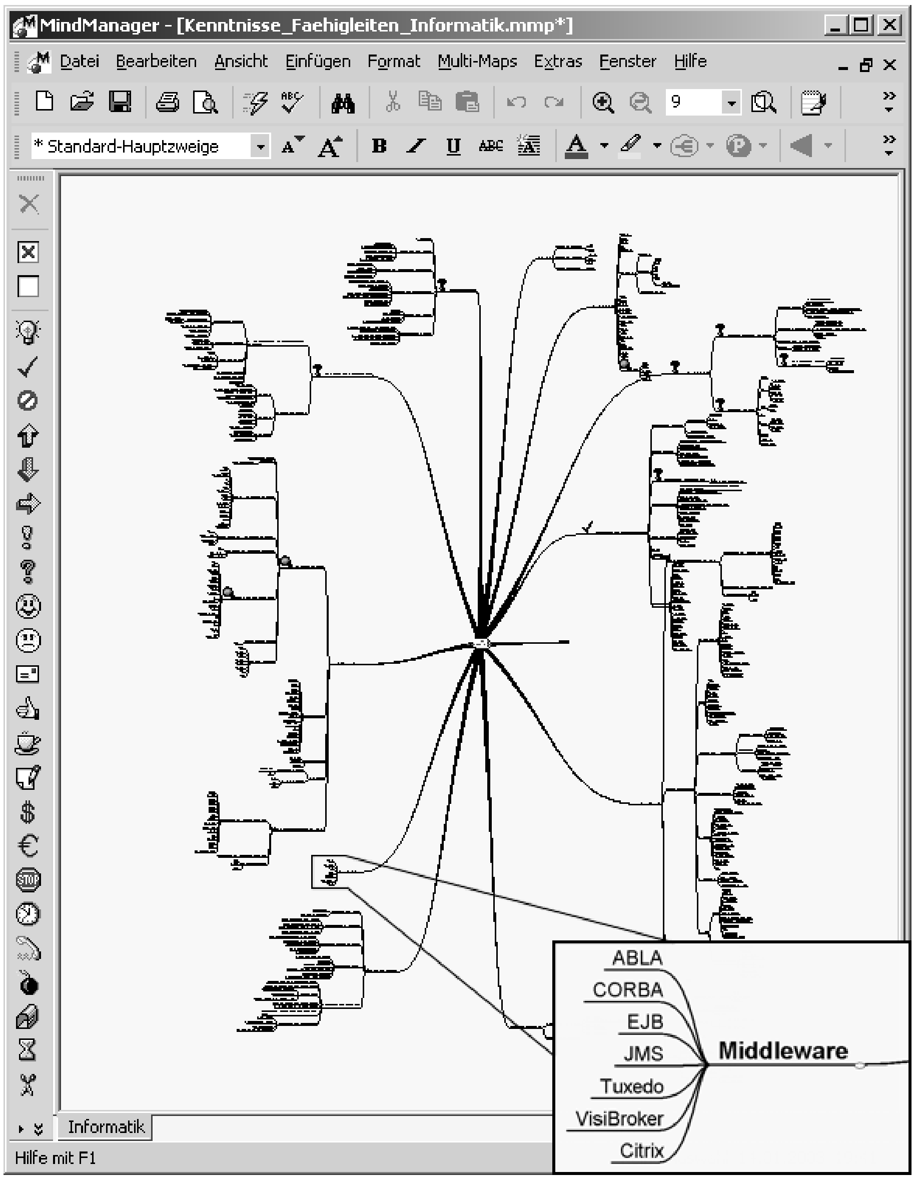


Figure 11.19: Mind map of skills @ Swiss Life

and a later formal evaluation of the skills ontology according to the proposed OntoClean Methodology possibly would have resulted changes of the ontology.

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? During the formalization we encountered several rather typical problems in ontology engineering like the mentioned ones. We will now present some of our design decisions that were taken.

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 kickoff phase. Due to different backgrounds and mind models of the developers they used different abstraction levels. This made it in the beginning rather difficult to merge the three models 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 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 were 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 were the high time consuming for finding concepts deeply into the structure and the amount of possible concepts, which can be chosen. The latter reason is based on the integrated skills ontology of all 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 was available at that time. An initial approach is presented in (Volz et al., 2002; 2003).

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 feel comfortable with a small ontology.

Concepts *vs.* Instances

To draw the borderline between concepts and instances can be a tricky task, as already known from expert systems. Though experienced knowledge engineers with a practical background prefer not to distinguish between concepts and instances during the modelling process at all (we refer to, *e.g.* Hans Akkermans from the Free University of Amsterdam, who was pleading for that), our underlying architecture made it necessary to make this distinction upfront (meanwhile an initial approach for solving such problems has been presented in (Motik et al., 2002)). As mentioned above it is difficult to distinguish between leaf nodes of the concept hierarchy and instances.

As a rule of thumb we decided to count the possible instances and decide if this would be enough to justify a concept or an instance. 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. This decision is also dependent on 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.

This problem does not occur when browsing the ontology, because super-concepts remain visible. But when users select an element as a skill then just "basics" occurs in his homepage, *i.e.* de-contextualization 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.

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.

11.4.5 Evaluation, Application & Evolution

As mentioned before, the case study was not carried out as it was planned. In particular the evaluation phase could not be carried out. Still, we considered the following aspects for the evolution of our skills management application.

Competencies and skills from employees evolve over time. 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 they are likely to be suggested by employees that are searching for particular skills and cannot find them in the current ontology. Suggestions should include both, the new skill itself as well as the position in the skills tree where it should be placed. To guarantee the quality of the ontology we planned the following evaluation strategy: While employees should suggest new skills, the experts should decide which skills should change in name and/or position in the skills tree and, additionally, decide which skills 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 appeared even in the same branch. For each domain there should exist a designated ontology manager who decides if and how suggested skills were finally integrated.

11.4.6 Main Lessons Learned @ Swiss Life

Complexity of Ontologies needed

According to the insight we have gained at Swiss Life during our participation in the On-To- Knowledge project Knowledge Management applications typically need ontology languages with a medium degree of expressiveness, reaching from simple taxonomies to hierarchies of concepts with properties and semantic relationships between concepts, including simple cardinality constraints. The more complex ontologies mainly result from the integration with other application systems. For example, the Skills Management application (SkiM) only uses a simple taxonomy for the available skills. However, to realize the application logic a more complex ontology lies underneath, although the user will never see it (*cf.* Figure 11.11). An extension of SkiM to cover functionalities like gap analysis, the integration with career planning and keeping track of attended training courses would require an additional degree of detail in the ontology.

Lessons Learned from Applying the Methodology

We experienced that the creation of an ontology is in its early phases a brainstorming activity and not a pure engineering task. Hence, brainstorming tools (*e.g.* like the used Mind Manager) are valuable add-ons that support the early stages of ontology engineering. The main purpose of these tools is to support the quick and intuitive capturing of knowledge. Especially domain experts not familiar with modelling issues felt comfortable with this tool, in fact they even suggested to use the Mind Manager due to the fact that they were familiar with it. However, at a certain point the process gets less and less brainstorming-driven and more construction-oriented. At Swiss Life, ontology modelling was done directly by a group of domain experts who have been guided and supported by experts in ontology design. In our experience, the domain expert become lost how to structure the ontology when it gets too big or has been changed around too often, because then they loose their feeling of directedness. It becomes more and more difficult to decide where to put which concepts and how to divide the ontology into substructures.

The (not surprising) conclusion is that ontology modelling is a design task which requires certain skills of its own. Ideally, the domain experts have those skills themselves. As such experts rarely exist (yet) the only way to go is to support domain experts by people who have those skills (as we did at Swiss Life). However, the interaction between the domain experts and the ontology design experts is difficult because there is nearly no common ground upon which they can build a common understanding: the domain experts do not understand the design issues (including especially too sophisticated design guidelines), the design experts do not understand the domain (in Swiss Life we had the lucky situation that one domain was IT where the design experts where also domain experts).

We dealt with this situation by up-skilling the domain experts with some basic modelling knowledge. We held several workshops, in the beginning to train them and then to support them during the more advanced modelling phases that were more engineering oriented. Within two weeks they were able to model a first skills hierarchy including over 700 skills from three different domains by themselves. Approximately six more weeks were needed to restructure their initial versions in collaboration with modelling experts into an appropriate hierarchy. This was due to time constraints of the participating domain experts. It has to be mentioned that all domain experts were highly self-motivated which certainly had a significant influence on the results produced.

11.5 Community of Knowledge Sharing @ BT

11.5.1 Introduction

The case study (*cf.* (Davies et al., 2003)[26]) on "Communities of Knowledge Sharing"[27] was carried out within BT's Research and Development organisation – BTexact Technologies.

BT (United Kingdom) is a leading company on the telecom market. BTexact Technologies is the subdivision of BT 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.

We now describe how we instantiated in close cooperation with our project partner BT the On-To-Knowledge Methodology (presented in Part II) in the case study.

11.5.2 Feasibility Study

Problems & Opportunities

The notion of communities of practice (*cf.* (Seely-Brown & Duguid, 1991)) has attracted much attention in the field of knowledge management. Communities of practice are groups within (or sometimes across) organizations who share a common set of information needs or problems. They are typically not a formal organizational unit but an informal network, each sharing in part a common agenda and shared interests or issues. In one example it was found that a lot of knowledge sharing among copier engineers took place through informal exchanges, often around a water cooler. The trends towards flexible working and globalization has led to interest in supporting dispersed communities using Internet technology (*cf.* (Davies, 2000)).

The challenge for organizations is to support such communities and make them effective. Provided with an ontology meeting the needs of a particular community of practice, knowledge management tools can arrange knowledge assets into the predefined conceptual classes of the ontology, allowing more natural and intuitive access to knowledge. Knowledge management tools must give users the ability to organize information into a controllable asset. Building an intranet-based store of information is not sufficient for knowledge management; the relationships within the stored information are vital. These relationships cover such diverse issues as relative importance,

[26]Further information on the BT case study can be found in the On-To-Knowledge deliverables (Krohn & Davies, 2001; Duke & Davies, 2002b; 2002a)

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

context, sequence, significance, causality and association. The potential for knowledge management tools is vast; not only can they make better use of the raw information already available, but they can sift, abstract and help to share new information, and present it to users in new and compelling ways.

Focus of KM Application

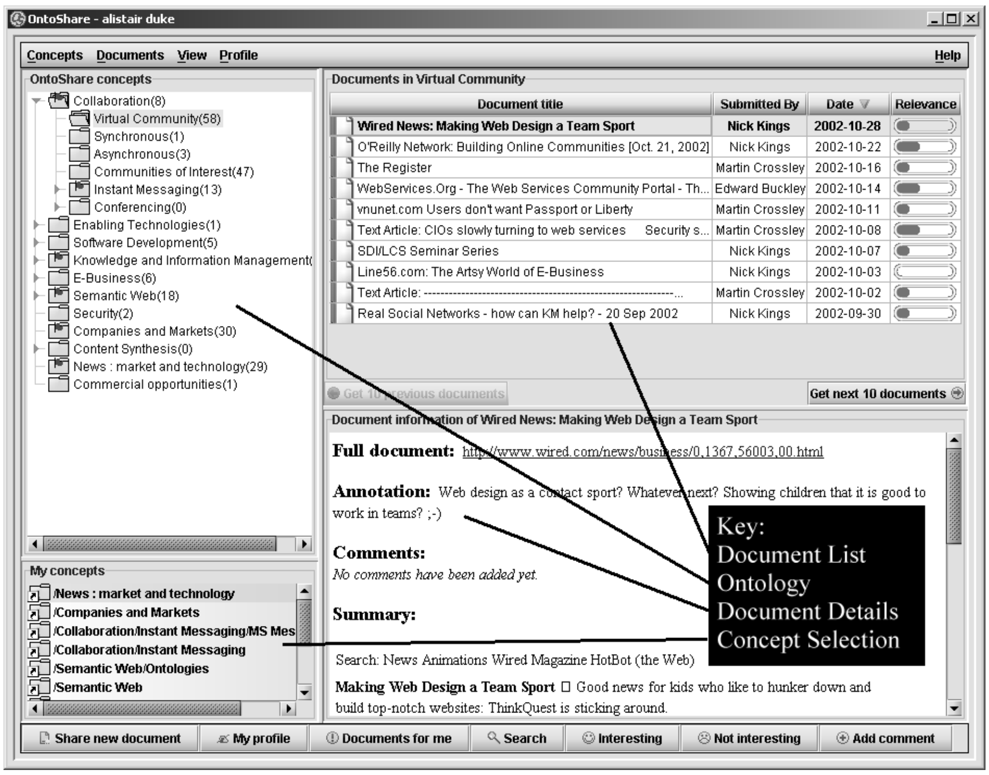
The case study is centered around the OntoShare system which facilitates and encourages the sharing of information between communities of practice within (or perhaps across) organizations and which encourages people – who may not previously have known of each other's existence in a large organization – to make contact where there are mutual concerns or interests. As users contribute information to the community, a knowledge resource annotated with metadata is created. Ontologies are defined using RDF Schema (RDFS) and populated using the Resource Description Framework (RDF).

We will briefly summarize the functionality of OntoShare (Figure 11.20 shows a screenshot of the OntoShare system). It is an ontology-based WWW knowledge sharing environment for a community of practice that models the interests of each user in the form of a user profile. In OntoShare, user profiles are a set of topics or ontological concepts (classes declared in RDFS) in which the user has expressed an interest. OntoShare has the capability to summarize and extract key words from WWW pages and other sources of information shared by a user and it then shares this information with other users in the community of practice whose profiles predict interest in the information. OntoShare is used to store, retrieve, summarize and inform other users about information considered in some sense valuable by an OntoShare user. This information may be from a number of sources: it can be a note typed by the user him/herself; it can be an intra/Internet page; or it can be copied from another application on the user's computer.

The goal of the case study is to introduce the newly developed ontology based OntoShare system as a successor of the pre-existing knowledge sharing system. As a major part this case study therefore includes a user-focussed evaluation of OntoShare.

People

The user group for the study consisted of approximately 30 researchers, developers and technical marketing professionals from the research and development arm of a large telecommunications firm. The interests of the users fell into 3 main groupings, similar to the domains of the domain experts: conferencing, knowledge and information management and personalization technologies. It was felt that three separate yet



Document title	Submitted By	Date	Relevance
Wired News: Making Web Design a Team Sport	**Nick Kings**	**2002-10-28**	●
O'Reilly Network: Building Online Communities [Oct. 21, 2002]	Nick Kings	2002-10-22	●
The Register	Martin Crossley	2002-10-16	●
WebServices.Org - The Web Services Community Portal - Th...	Edward Buckley	2002-10-14	●
vnunet.com Users don't want Passport or Liberty	Martin Crossley	2002-10-11	●
Text Article: CIOs slowly turning to web services Security s...	Martin Crossley	2002-10-08	●
SDI/LCS Seminar Series	Nick Kings	2002-10-07	●
Line56.com: The Artsy World of E-Business	Nick Kings	2002-10-03	
Text Article: --...	Martin Crossley	2002-10-02	●
Real Social Networks - how can KM help? - 20 Sep 2002	Nick Kings	2002-09-30	●

Document information of Wired News: Making Web Design a Team Sport

Full document: http://www.wired.com/news/business/0,1367,56003,00.html

Annotation: Web design as a contact sport? Whatever next? Showing children that it is good to work in teams? ;-)

Comments:
No comments have been added yet.

Summary:

Search: News Animations Wired Magazine HotBot (the Web)

Making Web Design a Team Sport □ Good news for kids who like to hunker down and build top-notch websites: ThinkQuest is sticking around.

Figure 11.20: Communities of knowledge sharing with OntoShare @ BT

overlapping topic areas would constitute an interesting mix of interests for the purposes of the trial.

11.5.3 Kickoff & Refinement

Since Kickoff and the Refinement actually were highly connected, we present them in a single section. The majority of the ontology development was carried out at a workshop held at the case study company's premises and run by a knowledge engineer. A selection of 6 key people from the user group were invited to attend the workshop. It was felt that as a whole, they would be able to cover the domain of interest for the whole of the user group. The workshop included a presentation that described in basic terms what an ontology is and how ontologies can be applied.

It also included a demonstration of the OntoShare tool which introduced the tool to the users as well as showing the use of an ontology in a practical application. Following this, the ontology development took place. Similar to the Swiss Life case study (*cf.*

Section 11.4), it was very much brainstorming oriented during the kickoff. The resulting semi-formal ontology description in form of a mind map™ is depicted in Figure 11.21.

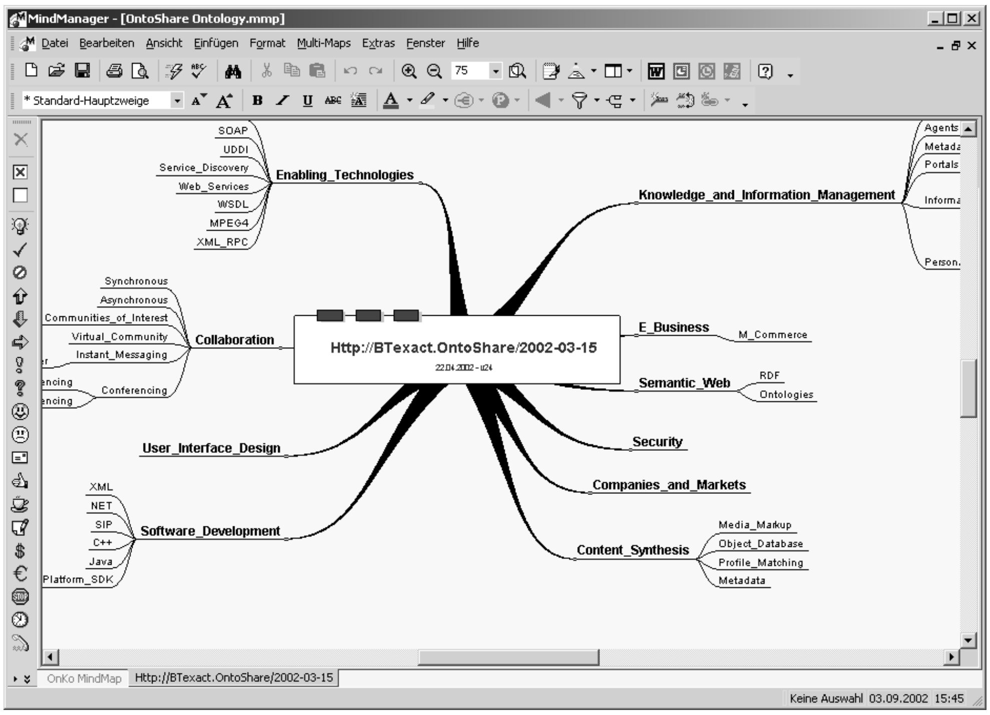


Figure 11.21: Mind map @ BT

The following is a description of the steps that were carried out subsequently.

1. A brief discussion took place on the number of concepts that was felt to be appropriate. Two suggestions of 20 and 50 were made. These were made in the context of the OntoShare tool and how its interface dealt with the ontology.

2. Each participant was asked to come up with 5+ topics that they felt were important to them. These were then collected. Some early organisation of the hierarchy took place at this stage.

3. Further organisation into a hierarchy then took place. This included adding and removing concepts and moving whole sub-branches of the proposed ontology.

4. The depth of the hierarchy was considered to be too deep. The UI aspects were considered at this stage and a decision taken to restrict the depth to a maximum of three levels where possible.

5. It was remarked that organizational groupings (*i.e.* knowledge management, conferencing, *etc.*) had been introduced at the top level of the hierarchy. It was decided that this might introduce unwanted boundaries for the users and that they might feel that they could only add documents to their particular part of the hierarchy. Also, some of the sub-branches straddled the top-level areas. The top-level was then removed.

6. The number of subconcepts per concept was considered in terms of the UI. It was felt that a maximum of 10-15 subconcepts would be manageable by the user. Concepts of minor importance were then removed or combined at this stage in order to achieve this.

7. The suitability of the overall ontology was then considered and a few additional refinements were made. The resulting ontology contained 10 top-level concepts and a total of 52 concepts with a maximum depth of three levels.

The group was able to produce this ontology at the workshop which meant that most of the refinement stage (*i.e.* the organization of concepts and relationships) had been carried out in tandem with the kickoff stage.

The over-riding lesson learned here is that user groups such as the one in this trial can be expected to produce lightweight ontologies.

11.5.4 User-Focused Evaluation

Of the three forms of evaluation in the On-To-Knowledge methodology, the most appropriate for use in the case study is user-focused. The tools rely on a high degree of user interaction and as such the users are the best resource for determining whether they meet their objectives. Various user-focused evaluation methods were employed. This section will describe these and the rationale for their use. The objectives of the evaluation were to determine:

- what the users think of sharing knowledge in an environment such as that used in the case study;
- whether the use of an ontology helps with the storing and sharing of knowledge
- whether the ontology evolution process is effective;
- whether the ontology developed as part of the case study was effective; and
- the good and bad points of the knowledge sharing environment.

The principal means of evaluating the views of the users was with the use of questionnaires. Questionnaires have the benefit of allowing the views of a high proportion of the users to be canvassed without burdening them a great deal. The questionnaires consisted of a mixture of open questions that required a qualitative response and a series of statements that required a quantitative response indicating the level to which the respondent agreed or disagreed. This mixed approach is endorsed by Eason who states:

> *"Structured questions have the virtue of easy analysis and direct comparability. Their weakness is that they pre-define the answers it is possible to give and may not therefore permit the user to report the most important issues. We have always found it useful to use a structured approach to reveal issues and, once an issue is located, to use an unstructured method to explore the nature of the issue."*
> (Eason, 1988)

A 'pre-trial' questionnaire and a 'post-trial' questionnaire were developed. The 'pre-trial' questionnaire was intended to determine the nature of the users in the case study in terms of the way (and how often) they access, receive and share information. The 'post-trial' questionnaire was intended to extract the user's views of and experiences with the OntoShare system.

Particular focus was placed upon the users' views on the usage of an ontology within the tool and the evolution of that ontology. An additional form of evaluation involved the analysis of usage statistics that were collected by an OntoShare module developed exactly for this purpose. This was able to record every interaction that occurred on the OntoShare server along with the user who performed it. This allows analysis to take place on the use of different OntoShare functions by the group as a whole as well as the behavior of individuals (which can then be cross-referenced with the questionnaire responses).

The combination of methods should allow an evaluation of the individual OntoShare functions to be made. The usage of the ontology can also be analyzed by recording the distribution of documents added to the concepts in the ontology and the evolution of the ontology over the course of the case study period. The final form of evaluation was an expert usability analysis. This involved an assessment of the OntoShare user interface and allowed a thorough inspection to be made by an independent expert. It generally results in a more objective and far reaching analysis than would be the case if it was carried out by someone connected to the development of the tool. The detailed results of the evaluation can be found in (Duke & Davies, 2002a). In Section 11.5.6 we will mention the most important results of the evaluation.

11.5.5 Application & Evolution

When a user shares some information in OntoShare, the system will match the content being shared against each known concept in the community's ontology. Each ontological concept is characterized in OntoShare by a set of terms (keywords and phrases). Following the matching process, the system suggests to the sharer a set of concepts to which the information could be assigned. The user is then able to accept the system recommendation or to modify it by suggesting alternative concept(s) to which the document should be assigned.

It is at this point that an opportunity for ontology evolution arises. Should the user indeed override the system's recommended classification of the information being shared, the system will attempt to modify the ontology to better reflect the user's conceptualization, as follows. The system will automatically extract the keywords and key phrases from the information given by the user. The set of such words and phrases are then presented to the user as candidate terms to represent the class to which the user has assigned the information. The user is free to select zero or more terms from this list and/or type in words and phrases of his own. The set of terms so identified is then added to the set of terms associated with the given concept, thus modifying its characterization.

We call this approach usage-based ontology evolution and in this way the characterization of a given concept evolves over time, this evolution being based on input from the community of users. We believe that this ability to change as users' own conceptualization of the given domain changes is a powerful feature which allows the system to better model the consensual ontology of the community.

Clearly, this level of evolution is limited to changing the semantic characterization of ontological classes and does not support, for example, the automatic suggestion of new concepts to be added to the ontology. More advanced ontology evolution is the subject of ongoing research (*cf.*, *e.g.*, (Stojanovic et al., 2002a)) .

We have seen above how users also indirectly annotate the information as a side-effect of sharing it with the community and we discuss and motivate this approach below. Pragmatically speaking, it is the case at the time of writing that only a very small proportion of WWW- and intranet-based information resources are annotated with RDF (meta)data. It is therefore beneficial to provide a system wherein such annotation effectively occurs as a side-effect of normal usage.

Another important observation is that it is in the general case impossible to cover the content of a document exhaustively by an RDF description. In practice, RDF descriptions can never replace the original document's content: any given RDF description of a set of resources will inevitably give one particular perspective on the information described. Essentially, a metadata description can never be complete since all possible uses for or perspectives on data can never enumerated in advance. Our approach accommodates this observation however in the sense that each community will create its own set of metadata according to its own interest in and perception of information that is added to its storage facility. It is very possible that the same information could be shared in two separate communities and emerge with different metadata annotations in each.

11.5.6 Main Lessons Learned @ BT

This section will suggest and discuss a number of recommendations for the evolution of the case study tools and deployment process. A set of lessons learned from the experience of carrying out the case study is presented. Firstly, recommendations concerned with improving OntoShare are considered. These have come to light as a direct result of the user-focused evaluation exercise.

- **Give careful consideration to the nature of the virtual community.** The evaluation showed that the majority of users were passive in their use of OntoShare *i.e.* they were happy to receive e-mails and read documents but did not add items to the system. As a result, the experience of the majority of OntoShare users is determined by the actions of the few who actually add items to the system. If those few users did not exist, the knowledge sharing benefits would not be forthcoming. Depending on the local organizational culture, dependence on a relatively small proportion of the user community may or may not be appropriate. Many successful communities are of this nature but in some cases alternative strategies may be required to reduce the dependence upon active users. In their responses to questionnaires, users made some useful suggestions in this regard. One user made the suggestion that OntoShare needed to be "regularly seeded with potentially relevant information or gain critical mass usage to offer a positive benefit and justify the effort of maintaining profiles and entering articles or information". This could be carried out manually by a knowledge engineer or automatically by an information agent and could benefit the system by ensuring sufficient data was added that was of interest to a wider cross section of the users. Both methods have drawbacks in that the manual process is time-consuming and the automatic method introduces the risk of downgrading the quality of information that is shared. The recommendation is to experiment with a combination of user, knowledge engineer and agent added data that can be varied depending upon user input and the nature of the community.
- **Provide better interface access.** A commonly occurring criticism of the system was its use of a Java applet to provide the interface. This proved to be slow to load and required a login step in order that the user could be recognized.

It was originally preferred instead of a HTML-based interface because displaying the Ontology (with a collapsing folder structure) would have proved difficult in that format. An alternative would be to provide an application, however this needs to be installed and re-installed every time a change is required which would have been disruptive for a system in its infancy. Also, different versions are required for different operating systems. The Java applet was so unpopular that using one of the alternatives now seems more attractive. The use of JavaScript and DHTML should be considered to allow the interface to operate in a standard browser. If this is not appropriate then the use of an installed application would probably be the best course of action.

- **Provide wider access to functions.** Another drawback with the system that was widely mentioned was the need to login to the system to provide comments, add items, etc. Alternative methods of accessing individual functions of OntoShare should be explored. These might include direct links in notification e-mails to a comment adding facility and support in a web browser for dropping a URL into the system. The intention should be to reduce the effort that is required to use each function.
- **Use richer ontological representation.** The OntoShare ontology currently contains concept relationships of the topic/subtopic type. Other relations are inferred by OntoShare and the user can request to see these. A better way of presenting these to the user is required. This might be a toggle where the relationships are indicated on screen when it is turned on.
- **Provide better support to new users.** When users first login they are often daunted by the interface. Better support should be provided to help them set up their profile and gain familiarity with the available functions. This could be in the form of a 'splash screen' that is shown on the first use of the tool or that can be disabled once users become familiar. This would show tips on usage and a short description of each function.
- **Inform users about an ontology change.** When changes to the ontology are accepted or rejected by the users, they should be notified of the outcome and invited to adjust their profiles accordingly. This will ensure that users can gain access to items added to new areas without having to login and browse the concepts.

The following lessons in relation to the development of ontologies have been learnt:

- **Physical presence is required.** The approach used to produce a domain ontology *i.e.* a group of experts in a focused workshop led by a knowledge engineer who is physically present, proved to be fruitful. The domain experts have limited time available, hence it is necessary to be very focused. Had this capture been carried out over a period of time involving a number of disparate people it would have probably been a drawn out process, lacking in focus.

- **Domain experts can be expected to produce a taxonomy.** A simple ontology in the form of a taxonomy is the most likely outcome from a group of domain experts asked to contribute in this way. More complex ontologies require considerably more effort. The OntoShare system is suited to a simple ontology with topic / subtopic relations.

- **The methodology provides an effective framework for the introduction of an ontology-based application.** The application of the methodology to the case study resulted in the rapid development of an ontology that performed well in its intended application. Users reported that they found it to be an appropriate ontology for their domains despite the limited amount of time that was available for ontology development.

In terms of the evaluation objectives introduced in Section 11.5.4, the following can be stated:

- Users are generally happy to receive shared knowledge from OntoShare and will often read it if it appears interesting and they have enough time. Only a minority of users actively share documents in OntoShare (mainly due to time pressures) so steps should be taken to make it as easy as possible to share. Data shared by users should be augmented with data added by a Knowledge Engineer or an information agent.

- Users reported that they found it useful to have the ontology available both when browsing for items and when adding items to the system.

- The ontology evolution process proved to be effective. Users were happy to create new concepts and these were accepted by the other users. Users reported that they found this facility useful although they were less inclined to make use of the interesting / not interesting features to evolve individual concept characterisations. Better access to facilities such as these has been recommended in order to further their usage.

- The ontology that was developed proved to be effective in the case study. It performed well in its intended application - measured by the distribution of documents that were added to its concepts. Users reported that they found it to be an appropriate ontology for their domains of interest.

The results of this evaluation will be used in the continued development and exploitation of OntoShare and related tools.

11.6 Virtual Organization @ EnerSearch

11.6.1 Introduction

EnerSearch is an industrial research consortium focused on IT and energy. Its aim is to create and disseminate knowledge on how the use of advanced IT will impact the energy utility sector, particularly in view of the fact that this industry branch is being liberalized across Europe. 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. Due to this wide geographical spread, EnerSearch has the character of a virtual organization also from the knowledge distribution point of view. In addition, for the general public interest it maintains a website[28] where it publishes many of its research results as papers, reports and books. Thus, dissemination of produced knowledge on IT and energy is a key function for EnerSearch. Within the On-To-Knowledge project, EnerSearch investigates whether Semantic Web methods and tools might be helpful to improve on this function, especially focused on its web information provisioning.

Goal of the case study (*cf.* (Sure & Iosif, 2002)[29]) on “Virtual Organization” 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 (*Quiz*RDF and Spectacle) *vs.* typical keyword based retrieval (EnerSEARCHer) during an experiment.

We now describe how we instantiated in close cooperation with our project partner EnerSearch the On-To-Knowledge Methodology (presented in Part II) in the case study.

[28] EnerSearch, see `http://www.enersearch.se/`

[29] Further information on the EnerSearch case study can be found in the On-To-Knowledge deliverables (Iosif et al., 2001; Iosif & Ygge, 2002; Iosif & Mika, 2002)

11.6.2 User-Focussed Evaluation

Since the ontology development was not within the main focus of the case study, we restrict our attention to the experiment about the user-focussed evaluation. In the following we describe how we designed the experiment and present some results. We start by naming potential threads for the experiment and how we face them. Next, we enumerate potential hypothesis for our experiment. Then we show the setting up of the experiment, including the selection of users and the technical solution. Last, but not least, we present some results of the experiment and conclude by a brief summary.

Potential Threads

We identified several potential threads that might affect the experiment. We list the threads and explain our strategy for avoiding them:

- **Users have not 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 has always been the 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 guides on how to use the tools were sent out in advance, in this way we minimised the chance of getting user problems with the actual test scenario.
- **Users are not interested:** By introducing the case study for our users before the actual test scenario we increased the interest for the project. The major part of the test users also had earlier experiences of the semantic web.
- **Too few users for a comparative study:** An important issue is the choice of subjects who are going to participate. Practical concerns often constraint 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 to 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 uses 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, *Quiz*RDF and Spectacle, we think that we have a good mix for

the case study. With the ontology based search tool , *Quiz*RDF, the user could start simple query consisting of only small number of search terms in order to get a picture on what kind of information that is available in the database. The second ontology based search tool, Spectacle, presents the information according to the inherent structuring that is offered by the ontology and thus 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 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 design 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.
- **System design:** The system must be logical and effective to use.

Hypotheses to be explored

In sum, in designing Semantic Web experiments different design dimensions are of importance: variations in information modes, in target user groups, and in individual information-processing styles (*cf.* (Iosif & Mika, 2002)). Any experiment must be based on one or more clearly formulated hypotheses that can be verified or falsified, for example by empirical-statistical methods. A possible list of testable hypotheses regarding Semantic Web-based information seeking is:

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.
3. The reduction in completion and number of mistakes will be more noticeable for less experienced users.
4. The reduction in time will also be more noticeable for users lacking a detailed knowledge of the underlying technical system implementation.

5. The ontology-based semantic access tools will be perceived as more useful than free text search by different types of persons for a broad range of domains and knowledge-acquisition scenarios.

6. The effort for developing and maintaining the ontology and information structure will not significantly exceed the effort to develop and maintaine the free text approach.

In a field experiment we tested hypotheses such as these for their significance. Furthermore we investigated how their validity varies with different information modes, target user groups, and individual information-processing styles.

Setting up the Experiment

The involvement of knowledge users in the experiment from the beginning is important because of the interaction between the users and tasks on the one hand, and their pre-knowledge, or lack thereof, of domain and/or systems on the other hand. We identified several types of target users for the tests through conducting a set of pre-trial interviews. As a result, the evaluation experiment includes three different types of interest groups.

One group consists of staff members from four different shareholder companies (the companies involved in the case study are: Sydkraft AB, Sweden, Iberdrola from Spain, Electricidade de Portugal, and ECN, The Netherlands). A second group consists of researchers from different scientific fields, several having at some time participated in EnerSearch projects. The third and final group intends to represent more or less a general outside audience and consists of students (studying at the department of software engineering and computer science at the Blekinge Institute of Technology in Sweden).

Finding information on a personal basis is important for all of these three groups, but for various reasons (that well reflect those found in geographically spread virtual organizations) they are generally limited in their time to invest for searching knowledge. The majority of the test users are familiar with the EnerSearch web and have used it before. There were of course also those who had never heard of the EnerSearch web but were introduced to and instructed on how the EnerSearch web is functioning.

We divided the total of 45 test users into six groups. We then mixed up the order of the questions for each group as well as the tool that each user group should use for answering each question, *i.e.* each group had three blocks – one block per available tool – with ten questions each. We now enumerate the questions that we posed to the test users:

1. Name a Knowledge Management methodology?
2. What approaches are used to describe business processes?
3. How large energy savings did the multi-agent system simulation of building control indicate?
4. When is it reasonable to adapt to changing communication conditions of a channel?
5. How does energy prices in Sweden compare to prices in Europe?
6. How does the liberation of an energy market influence the effectiveness of DSM measures?
7. How can you model the communication between intelligent agents on a society level?
8. How can spot pricing affect a distributor?
9. Name an organization that works with preparing protocol and equipment standards?
10. Who are the owners of EnerSearch?
11. What will happen to the price levels of electricity in Europe the next few years?
12. In what project has Claes Badenschneider been active?
13. Which are the three main mechanisms for a proper utility strategy according to Brousseau et al.?
14. What is the name of the building in Ronneby where a lot of field tests have been performed?
15. When did Hans Ottosson design and implement his first load management system?
16. To whom should a registration for ISPLC 2001 be sent?
17. What are the case studies in the On-To-Knowledge project?
18. What are the two technology-related problem areas where improvements are needed with respect to PLT?
19. To what e-mail address should general questions to EnerSearch be sent?
20. How can agents communicate with devices in an intelligent building?

21. What is life cycle cost and how to calculate it?
22. Which approach could be used to improve utility -customer relationship?
23. How does agent load management benefits the customer?
24. What styles of decision making could be distinguished?
25. What are some of the aspects to consider when assessing the costs of agent based load management?
26. How does load management effects pricing?
27. What are some of the weaknesses of the existing tools used to manage documents?
28. How is Information Technology changing the way organizations operate?
29. What is the name for an interorganizational relationship in which independent organizations share their resources, knowledge, costs and risks in order to produce a product?
30. Where can we find information on virtual organizations?

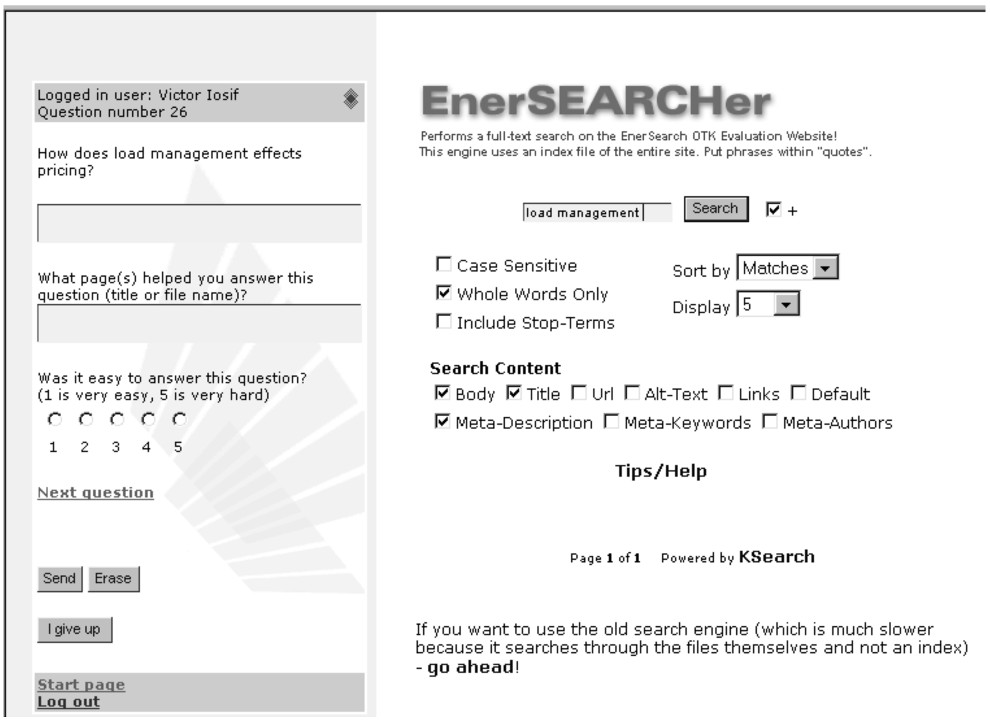
Please note that some questions were misleading or misspelled on purpose to check how fault-tolerant the used tools are.

The test persons used three different tools: the two ontology-based semantic tools *Quiz*RDF and Spectacle on the one hand and the free text search tool EnerSEARCHer on the other hand. The On-To-Knowledge tools *Quiz*RDF and Spectacle have the following advantages compared to a free text search tool like the EnerSEARCHER:

1. *Quiz*RDF has the advantage that the user can start with simple queries consisting of only small number of search terms in order to get a picture of what kind of information is available in the EnerSearch knowledge base. There is a continuum here from the common keyword-based search to different levels of semantic search.
2. The browsing tool Spectacle presents information according to the inherent structuring that is offered by an underlying domain ontology. This gives a valuable semantic context for the user not available through standard information retrieval tools. The information is presented in such a way that the path that leads to the information adds to the users' understanding of the semantic role of the information. Each concept chosen is "surrounded" by other ontologically similar concepts. Semantic generalizations and specializations (in different dimensions) are also offered to the user when browsing for information.

Technical Settings of the Experiment

The following Figures 11.22 – 11.25 show screenshots of all three tools during a user trial including the GUI that was used to guide users through the questionnaire during the experiment. The tools are all accessible with a typical browser.


Logged in user: Victor Iosif
Question number 26

How does load management effects pricing?

What page(s) helped you answer this question (title or file name)?

Was it easy to answer this question?
(1 is very easy, 5 is very hard)
○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Next question

Send Erase

I give up

Start page
Log out

EnerSEARCHer

Performs a full-text search on the EnerSearch OTK Evaluation Website!
This engine uses an index file of the entire site. Put phrases within "quotes".

load management | Search | ✓ +

☐ Case Sensitive
✓ Whole Words Only
☐ Include Stop-Terms

Sort by Matches
Display 5

Search Content
✓ Body ✓ Title ☐ Url ☐ Alt-Text ☐ Links ☐ Default
✓ Meta-Description ☐ Meta-Keywords ☐ Meta-Authors

Tips/Help

Page 1 of 1 Powered by KSearch

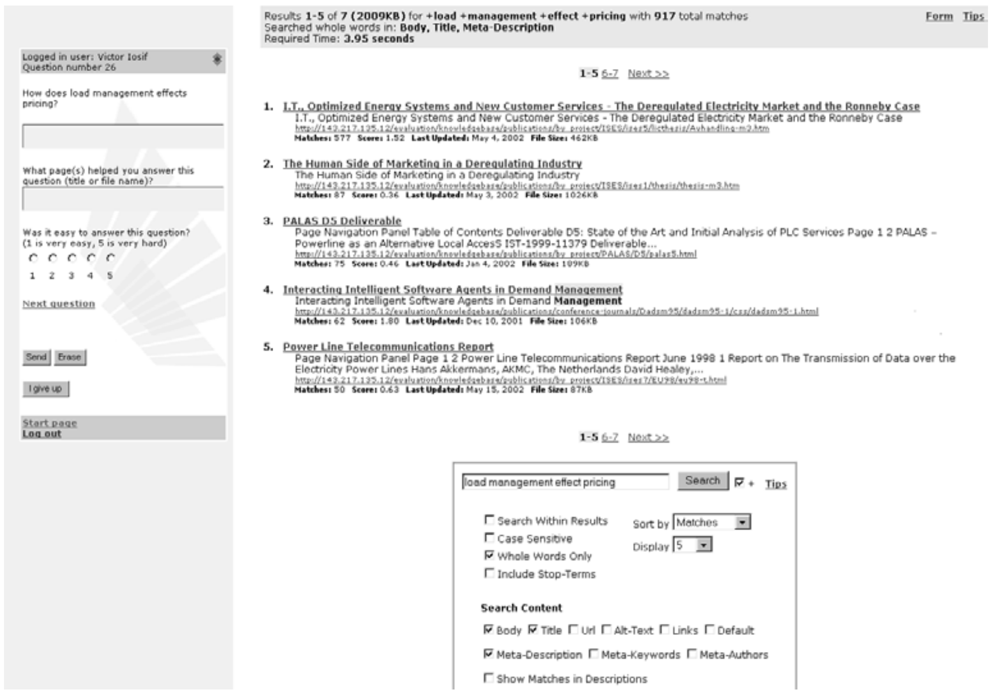
If you want to use the old search engine (which is much slower because it searches through the files themselves and not an index) - go ahead!

Figure 11.22: EnerSEARCHer (free text)

Figure 11.22 shows the setting during the experiment for the EnerSEARCHer. The screen is splitted into two parts, *viz.* frames.

On the left hand one sees a frame that is used to guide users through the questionnaire during the experiment. After a user is logged into the system, (s)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 (s)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 (s)he could push the "I give up button". Each user had to answer 30 questions, *i.e.* 10 with each tool. The questions were mixed up for different user groups.

On the right hand the currently active tool for answering a question (EnerSEARCHer, *Quiz*RDF 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 one sees the GUI of EnerSEARCHer, where one can easily recognize a typical query interface for keyword based search engines.



Logged in user: Victor Iosif
Question number 26

How does load management effects pricing?

What page(s) helped you answer this question (title or file name)?

Was it easy to answer this question?
(1 is very easy, 5 is very hard)

1 2 3 4 5

Next question

Send Erase

I give up

Start page
Log out

Results **1-5** of **7 (2009KB)** for **+load +management +effect +pricing** with **917** total matches
Searched whole words in: **Body, Title, Meta-Description**
Required Time: **3.95 seconds**

Form Tips

1-5 6-7 Next >>

1. I.T., Optimized Energy Systems and New Customer Services - The Deregulated Electricity Market and the Ronneby Case
   I.T., Optimized Energy Systems and New Customer Services - The Deregulated Electricity Market and the Ronneby Case
   http://143.217.135.12/evaluation/knowledgebase/publications/by_project/ISES/ises5/licthesis/Avhandling-m2.htm
   Matches: 577 **Score:** 1.52 **Last Updated:** May 4, 2002 **File Size:** 462KB
2. The Human Side of Marketing in a Deregulating Industry
   The Human Side of Marketing in a Deregulating Industry
   http://143.217.135.12/evaluation/knowledgebase/publications/by_project/ISES/ises1/thesis/thesis-m3.htm
   Matches: 87 **Score:** 0.36 **Last Updated:** May 3, 2002 **File Size:** 1026KB
3. PALAS D5 Deliverable
   Page Navigation Panel Table of Contents Deliverable D5: State of the Art and Initial Analysis of PLC Services Page 1 2 PALAS – Powerline as an Alternative Local AccesS IST-1999-11379 Deliverable...
   http://143.217.135.12/evaluation/knowledgebase/publications/by_project/PALAS/D5/palas5.html
   Matches: 75 **Score:** 0.46 **Last Updated:** Jan 4, 2002 **File Size:** 109KB
4. Interacting Intelligent Software Agents in Demand Management
   Interacting Intelligent Software Agents in Demand **Management**
   http://143.217.135.12/evaluation/knowledgebase/publications/conference-journals/Dadsm95/dadsm95-1/css/dadsm95-1.html
   Matches: 62 **Score:** 1.80 **Last Updated:** Dec 10, 2001 **File Size:** 106KB
5. Power Line Telecommunications Report
   Page Navigation Panel Page 1 2 Power Line Telecommunications Report June 1998 1 Report on The Transmission of Data over the Electricity Power Lines Hans Akkermans, AKMC, The Netherlands David Healey,...
   http://143.217.135.12/evaluation/knowledgebase/publications/by_project/ISES/ises7/EU98/eu98-t.html
   Matches: 50 **Score:** 0.63 **Last Updated:** May 15, 2002 **File Size:** 87KB

1-5 6-7 Next >>

load management effect pricing | Search | ✓ + | Tips

☐ Search Within Results
☐ Case Sensitive
✓ Whole Words Only
☐ Include Stop-Terms

Sort by: Matches
Display: 5

Search Content

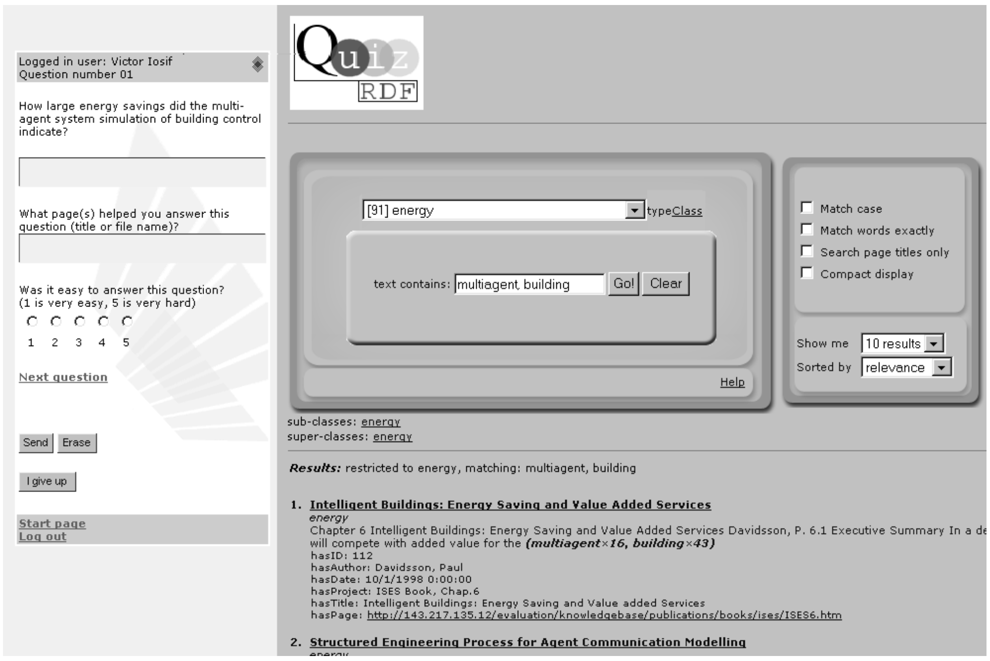
✓ Body ✓ Title ☐ Url ☐ Alt-Text ☐ Links ☐ Default
✓ Meta-Description ☐ Meta-Keywords ☐ Meta-Authors
☐ Show Matches in Descriptions

Figure 11.23: EnerSEARCHer results

In our first example screenshot Figure 11.22 the user “Victor Iosif” was currently trying to answer the question 26 “How does load management effects pricing?”. He had to answer the question with EnerSEARCHer. *E.g.* other users were given *Quiz*RDF or Spectacle. Figure 11.23 shows how EnerSEARCHer presents a result list corresponding to a query to the user (in a typical keyword based search engine like manner).

Figure 11.24 depicts the query interface of *Quiz*RDF. The user can start with entering keywords that (s)he is looking for – similar to a typical keyword based search like EnerSEARCHer. Additionally, (s)he can choose a concept (or “Class”) from a predefined ontology in the upper text box of *Quiz*RDF. *Quiz*RDF then returns documents that are relevant to the chosen concept and/or the given keywords. In the lower part a typical search result is presented. Additionally to the name of a document, *Quiz*RDF gives a short summary of a found document and metadata according to the predefined ontology.

Figure 11.25 illustrates the interface of Spectacle. On the left hand one sees an explorer-tree like browsing structure which is according to the underlying ontology. Spectacle offers the user some predefined views on the ontological structure, *e.g.* to start a search


Logged in user: Victor Iosif
Question number 01

How large energy savings did the multi-agent system simulation of building control indicate?

What page(s) helped you answer this question (title or file name)?

Was it easy to answer this question?
(1 is very easy, 5 is very hard)
1 2 3 4 5

Next question

Send Erase

I give up

Start page
Log out

[91] energy typeClass

text contains: multiagent, building Go! Clear

Match case
Match words exactly
Search page titles only
Compact display

Show me 10 results
Sorted by relevance

Help

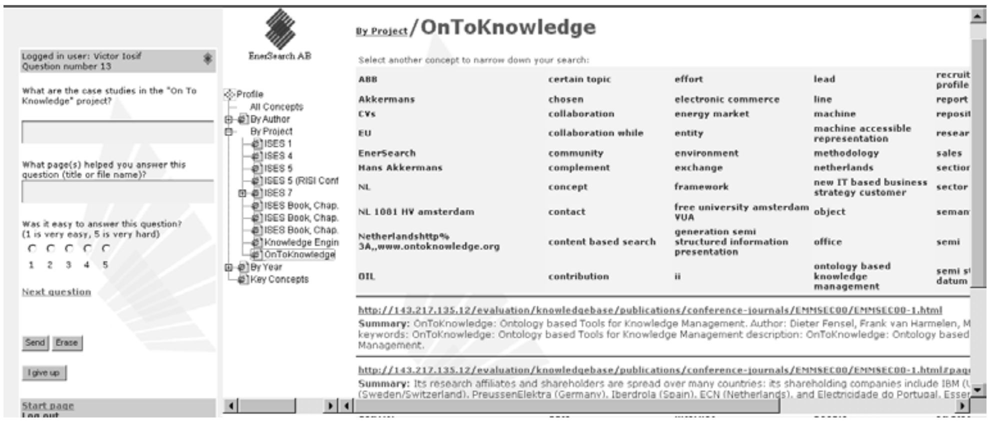
sub-classes: energy
super-classes: energy

Results: restricted to energy, matching: multiagent, building

1. **Intelligent Buildings: Energy Saving and Value Added Services**
   energy
   Chapter 6 Intelligent Buildings: Energy Saving and Value Added Services Davidsson, P. 6.1 Executive Summary In a de
   will compete with added value for the ***(multiagent×16, building×43)***
   hasID: 112
   hasAuthor: Davidsson, Paul
   hasDate: 10/1/1998 0:00:00
   hasProject: ISES Book, Chap.6
   hasTitle: Intelligent Buildings: Energy Saving and Value added Services
   hasPage: http://143.217.135.12/evaluation/knowledgebase/publications/books/ises/ISES6.htm

2. **Structured Engineering Process for Agent Communication Modelling**

Figure 11.24: *Quiz*RDF (ontology based)


Logged in user: Victor Iosif
Question number 13

What are the case studies in the "On To Knowledge" project?

What page(s) helped you answer this question (title or file name)?

Was it easy to answer this question?
(1 is very easy, 5 is very hard)
1 2 3 4 5

Next question

Send Erase

I give up

Start page
Log out

EnerSearch AB

Profile
All Concepts
By Author
By Project
ISES 1
ISES 4
ISES 5
ISES 5 (RISI Conf
ISES 7
ISES Book, Chap.
ISES Book, Chap.
ISES Book, Chap.
Knowledge Engin
OnToKnowledge
By Year
Key Concepts

By Project/**OnToKnowledge**

Select another concept to narrow down your search:

ABB	certain topic	effort	lead	recruit profile
Akkermans	chosen	electronic commerce	line	report
CVs	collaboration	energy market	machine	reposit
EU	collaboration while	entity	machine accessible representation	resear
EnerSearch	community	environment	methodology	sales
Hans Akkermans	complement	exchange	netherlands	sectior
NL	concept	framework	new IT based business strategy customer	sector
NL 1081 HV amsterdam	contact	free university amsterdam VUA	object	seman
Netherlandshttp%3A,,www.ontoknowledge.org	content based search	generation semi structured information presentation	office	semi
OIL	contribution	ii	ontology based knowledge management	semi st datum

http://143.217.135.12/evaluation/knowledgebase/publications/conference-journals/EMMSEC00/EMMSEC00-1.html
Summary: OnToKnowledge: Ontology based Tools for Knowledge Management. Author: Dieter Fensel, Frank van Harmelen, M
keywords: OnToKnowledge: Ontology based Tools for Knowledge Management description: OnToKnowledge: Ontology based
Management.

http://143.217.135.12/evaluation/knowledgebase/publications/conference-journals/EMMSEC00/EMMSEC00-1.html#page
Summary: Its research affiliates and shareholders are spread over many countries: its shareholding companies include IBM (U
(Sweden/Switzerland), PreussenElektra (Germany), Iberdrola (Spain), ECN (Netherlands), and Electricidade do Portugal, Esser

Figure 11.25: Spectacle (ontology based)

by looking for "Authors", "Projects" or "Year". These concepts were identified as an intuitive starting point for the search. Known Projects are shown as childs of the "Projects"–Node, *e.g.* the On-To-Knowledge project. On the right hand of the interface one can see on the upper side the currently chosen navigational path, *e.g.* "By Project/OnToKnowledge". Below the user gets presented the context of a chosen concept, *viz.* related concepts. Users may follow these links to narrow down their navigational search. At the bottom one can see a result list indicating the relevant documents for the currently chosen navigational path in the ontology.

Some Results

Each log entry of the experiment contains the following items: question number, answer (text), name of the user, time duration for answering the question, the user group 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"). In this analysis we concentrated on getting an impression for the hypothesis 1 and 2 (*cf.* Section 11.6.2), the complete results can be found in (Iosif & Mika, 2002).

Figure 11.26 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. 10,20% of the questions answered by using *Quiz*RDF were answered wrong, 57,14% were answered right and in 32,65% the user gave up. 23,73% of the questions answered by using Spectacle were answered wrong, 40,68% were answered right and in 35,59% cases the user gave up.

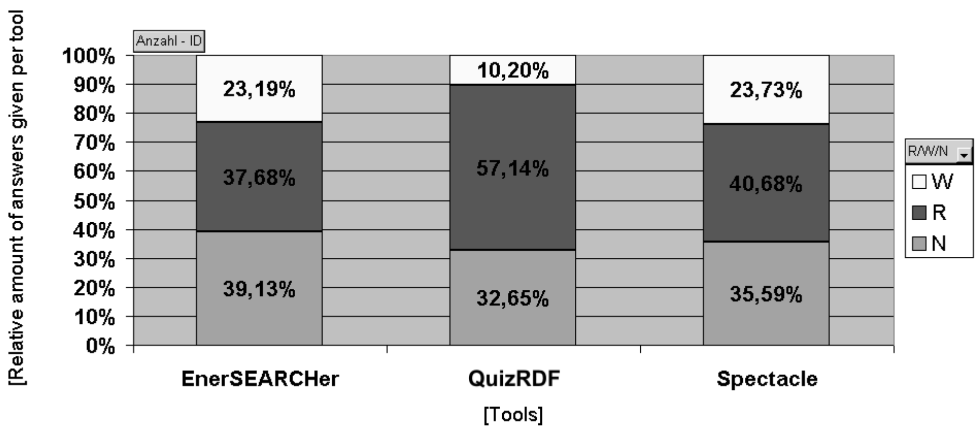


Figure 11.26: How relatively often did users give (W)rong, (R)ight or (N)o answers with each tool?

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

Figure 11.27 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 *Quiz*RDF (25,77%), followed by EnerSEARCHer (34,71%) and Spectacle (39,52%).

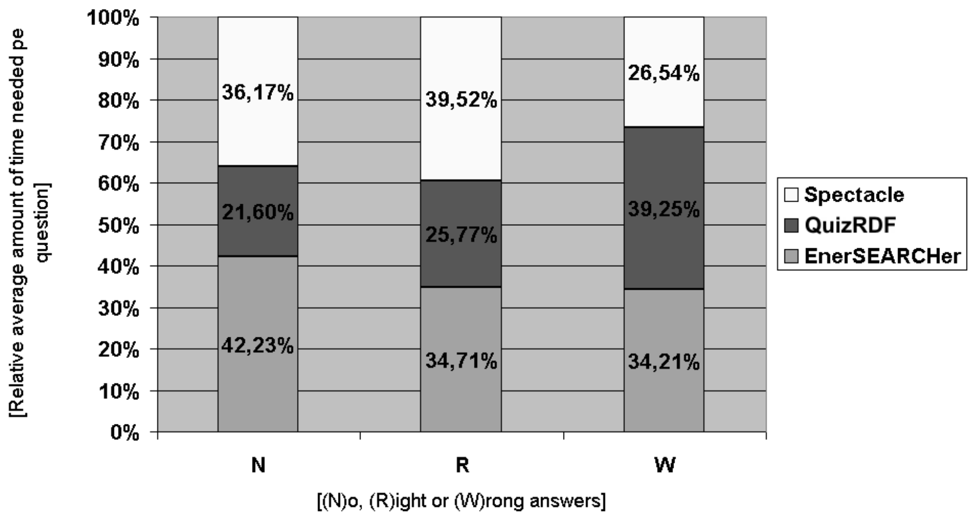


Figure 11.27: What relative average amount of time needed users for (W)rong, (R)ight or (N)o answering of one single question?

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

11.6.3 Main Lessons Learned @ EnerSearch

To demonstrate the real value of Semantic Web we need to carry out field experiments. We have outlined a number of hypotheses that we tried to answer to in our case study. We also have 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 information processing.

Our results indicate that

1. our **hypothesis 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”, **is supported by our results**, and
2. our **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”, **is partially supported by our result**.

As a general conclusion we summarize that the ontology based tools are generally at least as good as the keyword based tool and to some extent they are better. The complete results, *e.g.* including also the results for other hypotheses, can be found in (Iosif & Mika, 2002). This deliverable also contains a chapter on technology-focussed evaluation of the On-To-Knowledge tool suite.

11.7 Overall Lessons Learned in On-To-Knowledge

We conclude this chapter presenting the main lessons learned from applying the methodology in the case studies of On-To-Knowledge. Further information about the project can *e.g.* be found in the Final Project Report (Fensel et al., 2002).

- **Early ontology development** is often unstructured **brainstorming** rather than careful design. Such brainstorming is currently unsupported by many ontology engineering tools like Protégé or OilEd (*cf.* Section 13.2). We relied on an integrated a commercially successful and widely used brainstorming tool. We extended the functionalities of this tool to meet our requirements, in particular we connected the tool and OntoEdit via the plug-in Mind2Onto to close the gap between capturing and formalization of knowledge.
- **Different processes drive KM projects**, but “Human Issues” might dominate other ones (as already outlined by Davenport). We had to learn hard lessons there, two of the case studies were heavily affected by major internal restructurings of the companies, *viz.* the Swiss Life and the BT case studies.
- **Guidelines for domain experts in industrial contexts** have to be **pragmatic**, otherwise they are unlikely to be understood and to be used at all.
- **Collaborative ontology engineering** requires **physical presence *and* advanced tool support**. On the one hand we not only had to train, support and guide domain experts, but also to personally motivate them to contribute to our project. On the other hand they could work in the mean time by themselves on “their part of the ontology” due to the strong support of the tools.

12 OntoWeb

Overview

In **this chapter** we present work performed within the thematic network OntoWeb. In Section 12.1 we introduce the main goals of the network and present an overview of our contributions within the network. The focus in the following sections is on two topics: (i) the creation of the OntoWeb Semantic Portal on top of our conceptual architecture for SEmantic portALs (SEAL) (*cf.* Section 12.2) and (ii) the evaluation of ontology based tools in the workshop on "Evaluation of Ontology based Tools (EON)" (*cf.* Section 12.3).

References: This chapter is mainly based on (Studer et al., 2002), (Maedche et al., 2002b), (Sure, 2002c), and (Sure & Angele, 2002).

12.1 About the Thematic Network

The EU IST-2000-29243 "OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce" thematic network[1] is a part of the Information Societies Technologies (IST) Programme of the European Union (EU). The thematic network has currently over 100 partners coming from academia and industry. Most of them are located in Europe, but there exist also strong links to communities in the United States of America and Asia. The main objectives of OntoWeb are described in the OntoWeb Annex:

> *"The goal of OntoWeb Network is to bring researcher and industrials together enabling the full power ontologies may have to improve information exchange in areas such as: information retrieval, knowledge management, electronic commerce, and bioinformatics. It will also strengthen the European influence on standardization efforts in areas such as web languages*

[1] OntoWeb, see `http://www.ontoweb.org`

> *(RDF, XML), upper-layer ontologies, and content standards such as catalogues in electronic commerce."*
> (OntoWeb, 2001)

The OntoWeb Annex continues with the expected main results:

1. A technical roadmap on the state of the art of the WWW of the next generation plus guides to Industrial and Commercial Applications.
2. A series of European and international workshops that bring together leading researchers and industrials.
3. A web portal on advanced knowledge management and electronic commerce.
4. Contributions in content and language standardization.
5. A scientific journal and educational material.

Inherently the major goal of the thematic network OntoWeb is the dissemination of related research in numerous deliverables as well as workshop and conference proceedings. The work described in this thesis contributed mainly to the items 1, 2 and 3 as described in the following enumeration that is corresponding to the previous one.

1. The main Parts II, III and IV of this work, *i.e.* the On-To-Knowledge Methodology, OntoEdit and the case studies, have been contributed to a series of technical roadmap deliverables, *cf.* (Gómez-Pérez et al., 2002b; 2002a; Fernandéz-López et al., 2002; Léger et al., 2002a; 2002b).
2. As part of the activities in the "Special Interest Group on Tools (SIG4)" a workshop on "Evaluation of Ontology-based Tools (EON2002)" was held (*cf.* (Sure & Angele, 2002)). The workshop included an experiment for evaluating different state-of-the-art Ontology Engineering Environments. We describe the aims, results and lessons learned of the workshop in Section 12.3.
   Furthermore, OntoEdit's collaborative and inferencing capabilities (*cf.* Part III) have been presented at two OntoWeb sponsored conferences, *viz.* at the ISWC 2002[2] (*cf.* (Sure et al., 2002a))and the ODBASE 2002[3] (*cf.* (Sure et al., 2002b)).

[2]"First International Semantic Web Conference: The Semantic Web (ISWC 2002)", *cf.* (Horrocks & Hendler, 2002), see `http://iswc.semanticweb.org/`.

[3]"First International Conference on Ontologies, Databases, and Applications of Semantics for Large-Scale Information Systems (ODBASE 2002)", co-located with the "Confederated International Conferences: On the Move to Meaningful Internet Systems (CoopIS, DOA, and ODBASE 2002)", *cf.* (Meersman et al., 2002), see `http://www.cs.rmit.edu.au/fedconf/`.

3. A Semantic Portal for the OntoWeb Community has been established (*cf.* (Spyns et al., 2002b)) on top of the conceptual architecture SEAL for "SEmantic portALs" (*cf.* (Maedche et al., 2002b; Studer et al., 2002)). The work on the portal has also been described in a series of OntoWeb deliverables, *cf.* (Studer et al., 2001b; 2001a; Majer et al., 2002). This work was performed in collaboration with our partner, the StarLab at the Free University of Brussels headed by Robert Meersman. They provided the advanced browsing and querying facilities of the OntoWeb Semantic Portal based on the DOGMA Server (*cf.* (Meersman, 1999; Jarrar & Meersman, 2002; Spyns et al., 2002a)). A detailed description of the collaborative results can be found in (Spyns et al., 2002b). We illustrate the OntoWeb Semantic Portal in the following Section 12.2.

12.2 SEmantic portAL (SEAL) of OntoWeb

In this section we present the AIFB part of the OntoWeb Semantic Portal. Providing the complete picture (including the work of our project partner, the StarLab of the Free University of Brussels (VUB)) is beyond the scope of this work, a detailed description of the complete portal, including the advanced querying and browsing facilities provided by the DOGMA Server (*cf.* (Meersman, 1999; Jarrar & Meersman, 2002; Spyns et al., 2002a)) can be found in (Spyns et al., 2002b).

We start with a motivation for our architecture for Semantic Portals in Section 12.2.1 and show different aspects on web information integration in Section 12.2.2 and web site management in Section 12.2.3. Then we illustrate the implementation of the OntoWeb Semantic Portal in Section 12.2.4 with a focus on the underlying process model for publishing content in Section 12.2.5. We conclude by describing future directions for SEAL and OntoWeb in Section 12.2.6.

12.2.1 Motivation for Semantic Portals

The recent decade has seen a tremendous progress in managing semantically heterogeneous data sources. Core to the semantic reconcilation between the different sources is a rich conceptual model that the various stakeholders agree on, an *ontology.* The conceptual architecture developed for this purpose now generally consists of a three layer architecture comprising (*cf.* (Wiederhold, 1993; Wiederhold & Genesereth, 1997))

1. heterogeneous **data sources** (*e.g.*, databases, XML documents, but also data found in HTML tables),
2. **wrappers** that lift these data sources onto a common data model (*e.g.* OEM (Papakonstantinou et al., 1995) or RDF (Lassila & Swick, 1999)),

3. integration modules (**mediators** in the dynamic case) that reconcile the varying semantics of the different data sources.

Thus, the complexity of the integration/mediation task could be greatly reduced.

Similarly, in recent years the information system community has successfully strived to reduce the effort for managing complex web sites (Anderson et al., 1999; Ceri et al., 1999; 2000; Fraternali & Paolini, 1998; Fernandez et al., 2000; Mecca et al., 1999)). Previously ill-structured web site management has been structured with process models, redundancy of data has been avoided by generating it from database systems and web site generation (including management, authoring, business logic and design) has profited from recent, also commercially viable, successes (Anderson et al., 1999). Again we may recognize that core to these different web site management approaches is a rich conceptual model that allows for accurate and flexible access to data. Similarly, in the hypertext community conceptual models have been explored that im- or explicitly exploit ontologies as underlying structures for hypertext generation and use (Crampes & Ranwez, 2000; Rossi et al., 2000; Goble et al., 2001).

SEAL is our conceptual architecture for SEmantic PortALs that aims at facilitating the management of community web sites and web portals on an ontology basis. The ontology supports queries to multiple sources (a task also supported by semi-structured data models (Fernandez et al., 2000)), but beyond that it also includes the intensive use of the schema information itself allowing for automatic generation of navigational views[4] and mixed ontology and content-based presentation. The core idea of SEAL is that Semantic Portals for a community of users that contribute *and* consume information (Staab et al., 2000) require web site management *and* web information integration. In order to reduce engineering and maintenance efforts SEAL uses an ontology for semantic integration of existing data sources as well as for web site management and presentation to the outside world. SEAL exploits the ontology to offer mechanisms for acquiring, structuring and sharing information between human and/or machine agents.

The SEAL conceptual architecture (*cf.* Figure 12.1; details to be explained subsequently below) depicts the general scheme. Approaches for web site management emphasize on the upper part of the figure and approaches for web information integration focus on the lower part while SEAL combines both with an ontology as the knot in the middle.

The origins of SEAL lie in Ontobroker (*cf.* (Decker et al., 1999)), which was conceived for semantic search of knowledge on the Web and also used for sharing knowledge on the Web (*cf.* (Benjamins & Fensel, 1998; Benjamins et al., 1999)), also taking

[4]Examples are navigation hierarchies that appear as `has-part`-trees or `has-subtopic` trees in the ontology.

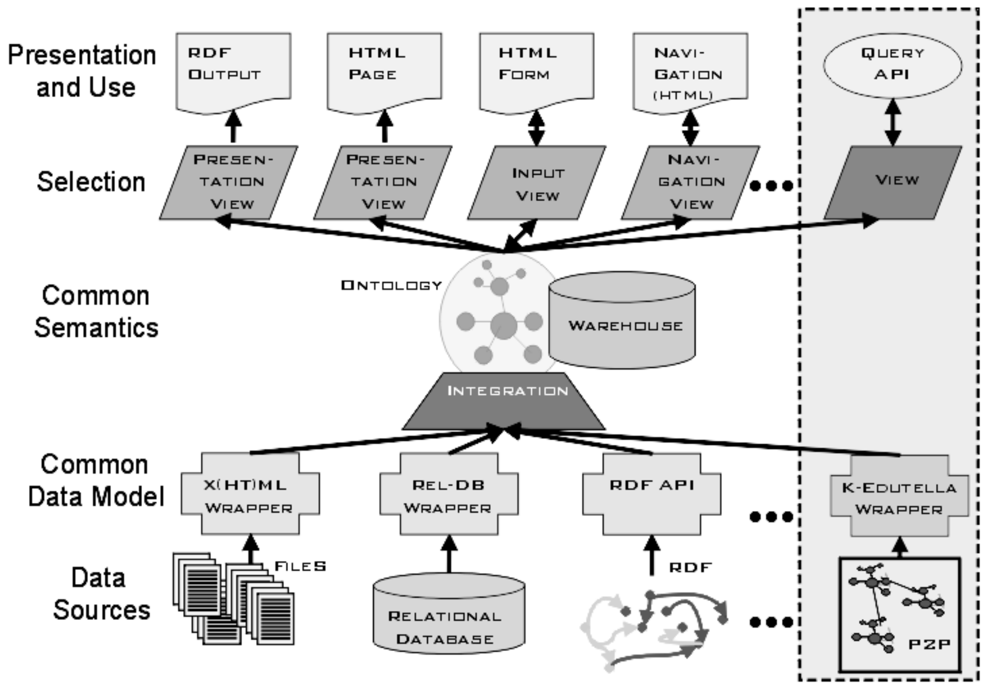


Figure 12.1: SEAL conceptual architecture

advantage of the mediation capabilities of ontologies (*cf.* (Fensel et al., 2000a)). It then developed into an overarching framework for search and presentation offering access at a portal site (*cf.* (Staab et al., 2000)). This concept was then transferred to further applications (*cf.* (Angele et al., 2000; Sure et al., 2000)) and constitutes the technological basis for several Semantic Portals, *viz.* (i) the "OntoWeb Semantic Portal"[5] (which is, as mentioned before, also based on the DOGMA Server approach of the StarLab, *cf.* (Spyns et al., 2002b)), (ii) the portal of the Institute AIFB that is also called "AIFB Semantic Portal"[6], and (iii) the "KA2 Community Web Portal"[7]. It now combines the roles of information integration in order to provide data for the Semantic Web and for a Peer-to-Peer network with presentation to human Web surfers.

12.2.2 Web Information Integration

One of the core challenges when building a data-intensive web site is the integration of heterogeneous information on the WWW. The recent decade has seen a tremendous

[5]OntoWeb Semantic Portal, `http://www.ontoweb.org`
[6]AIFB Semantic Portal, `http://www.aifb.uni-karlsruhe.de`
[7]KA2 Community Web Portal, `http://ka2portal.aifb.uni-karlsruhe.de`

progress in managing semantically heterogeneous data sources (Wiederhold & Genesereth, 1997; Fernandez et al., 2000). The general approach we pursue is to "lift" all the different input sources onto a common data model, in our case RDF(S) (*cf.* (Lassila & Swick, 1999; Brickley & Guha, 2002)). Additionally, an ontology acts as a semantic model for the heterogeneous input sources. As mentioned earlier and visualized in our conceptual architecture in Figure 12.1, we consider different kinds of data sources of the Web as input.

1. First of all, to a large part the Web consists of static HTML pages, often semi-structured, including tables, lists, *etc.* We have developed an ontology-based HTML wrapper that is based on a semi-supervised annotation approach. Thus, based on a set of predefined manually annotated HTML pages, the structure of new HTML pages is analyzed, compared with the annotated HTML pages and relevant information is extracted from the HTML page. The HTML wrapper is currently extended to also deal with heterogeneous XML files.

2. We use an automatic XML (*cf.* (Bray et al., 2000)) wrapping approach that has been introduced in (Erdmann & Studer, 2001; Erdmann, 2001). The idea behind this wrapping approach is that these XML documents refer to an DTD that has been generated from the ontology. Therefore we automatically generate a mapping from XML to our data model so that integration comes for free.

3. Data-intensive applications typically rely on relational databases. A relational database wrapping approach (Stojanovic et al., 2002c) maps relational database schemas onto ontologies that form the semantic basis for the RDF statements that are automatically created from the relational database.

4. In an ideal case content providers have been registered and agreed to describe and enrich their content with RDF-based meta data according to a shared ontology. In this case, we may easily integrate the content automatically by executing an integration process. If content providers have not been registered, but provide RDF-based meta data on their Web pages, we use ontology-focused meta data discovery and crawling techniques to detect relevant RDF statements.

12.2.3 Web Site Management

One difficulty of community portals lies in integrating heterogeneous data sources. Each source may be hosted by different community members or external parties and fulfills different requirements. Therefore typically all sources vary in structure and design. Community portals like (in our case) the web site of our own institute require coherence in hosted information on different levels. While the information integration aspect (see previous section) satisfies the need for a coherent structure that is provided by the ontology we will now introduce various facilities for construction and maintenance of web sites to offer coherent style and design. Each facility is illustrated by our conceptual architecture (*cf.* Figure 12.1).

- **Presentation view:** Based on the integrated data in the warehouse we define user-dependent presentation views. First, as a contribution to the Semantic Web, our architecture is dedicated to satisfy the needs of software agents and produces machine understandable RDF. Second, we render HTML pages for human agents. Typically *queries for content* of the warehouse (*cf.* Figure 12.1) define presentation views by selecting content, but also *queries for schema* might be used, *e.g.* to label table headers.
- **Input view:** To maintain a portal and keep it alive its content needs to be updated frequently not only by information integration of different sources but also by additional inputs from human experts. The input view is defined by *queries to the schema*, *i.e.* queries to the ontology itself. Similar to (Grosso et al., 1999) we support the knowledge acquisition task by generating forms out of the ontology. The forms capture data according to the ontology in a consistent way which are stored afterwards in the warehouse.
- **Navigation view:** To navigate and browse the warehouse we automatically generate navigational structures by using *combined queries for schema and content*. First, we offer different user views on the ontology by using different types of hierarchies (*e.g. is-a*, *part-of*) for the creation of top level navigational structures. Second, for each shown part of the ontology the corresponding content in the warehouse is presented. Therefore especially users that are unfamiliar with the portal are supported to explore the schema and corresponding content.
- **(General) View:** In the future it is planned to explore techniques of handling updates on these views. A first approach has been presented in (Volz et al., 2002; 2003).

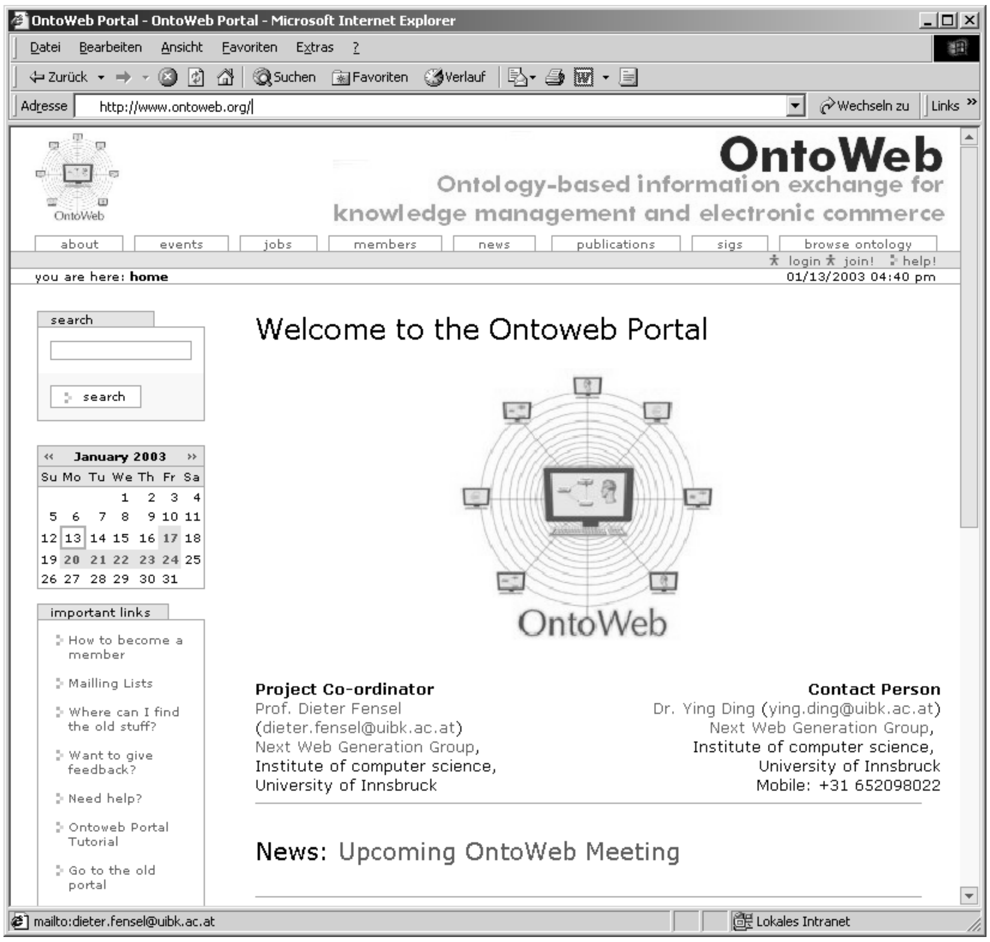
12.2.4 Implementation of the OntoWeb Semantic Portal

The OntoWeb Semantic Portal serves as a platform for the OntoWeb community members to disseminate their activities, events, publications and other related information. The general objective of the portal is described in the OntoWeb Annex:

> *"The network will provide a portal offering an integrated access to all kind of information related to the network. This portal will offer browsing as well*

as advanced querying facilities as provided by ontology-based approaches to community portals."
(OntoWeb, 2001)

The OntoWeb Semantic Portal (*cf.* Figure 12.2) combines the SEAL architecture with the DOGMA approach (*cf.* (Spyns et al., 2002b)) and as such it is structured according to an ontology which serves as a shared basis for supporting communication between humans and machines (*cf.* Section 3.3). Our approach aims at semi-automatical construction of a community portal using the community's meta data to enable information provision, querying and browsing of the portal.


OntoWeb Portal - OntoWeb Portal - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Verlauf
Adresse http://www.ontoweb.org/
Wechseln zu Links
OntoWeb
about events jobs members news publications sigs browse ontology
login join! help!
search
important links
OntoWeb
mailto:dieter.fensel@uibk.ac.at
Lokales Intranet

OntoWeb
Ontology-based information exchange for knowledge management and electronic commerce

you are here: **home** 01/13/2003 04:40 pm

«	January 2003					»
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

- How to become a member
- Mailling Lists
- Where can I find the old stuff?
- Want to give feedback?
- Need help?
- Ontoweb Portal Tutorial
- Go to the old portal

Welcome to the Ontoweb Portal

Project Co-ordinator
Prof. Dieter Fensel
(dieter.fensel@uibk.ac.at)
Next Web Generation Group,
Institute of computer science,
University of Innsbruck

Contact Person
Dr. Ying Ding (ying.ding@uibk.ac.at)
Next Web Generation Group,
Institute of computer science,
University of Innsbruck
Mobile: +31 652098022

News: Upcoming OntoWeb Meeting

Figure 12.2: The OntoWeb Semantic Portal

For this purpose we could reuse the SEAL architecture and the DOGMA approach, but we also had to provide facilities for content management. In particular, to ensure the content quality of the portal we added a process model for reviewing newly inserted content. The process model and its implementation in a publishing workflow are described in the next section.

The ontology is used for several purposes, as sketched in the following.

- **Portal generation & navigation**: The top level navigational structure of the portal is derived from the ontology, *i.e.* the portal itself is generated from the ontology. As indicated above, this allows for a flexible conceptual modelling of the portal instead of hardwired structure.
- **Acquisition of internal content:** Content can be directly feeded into the portal through automatically generated forms. The fields of the forms are derived from the ontology, *e.g.* for the concept Publication exists the relation HASAUTHOR with the range Person. When entering data about a new publication a specific form for Publication has therefore a field for capturing authors. This kind of knowledge acquisition was already successfully applied in the Protégé community[8] and is an integral part of Protégé 2000 (*cf.* (Noy et al., 2000)).
- **Integration of external content:** Each community member can provide meta data, *e.g.* meta data describing his organization, staff, research directions or publications. All meta data that are structured according to the OntoWeb ontology are transferred into the portal via syndication. The meta data are typically stored in the headers of HTML documents. The enrichment of documents with meta data is usually called annotation (see also Section 5.3).
- **Querying & browsing:** The query and browsing facility on top of DOGMA makes use of the ontology to provide an enhanced searching. Users not only can use keywords for queries, but they also can use the ontology itself to create queries. Typically the precision of the derived answers is much higher than in purely keyword-based retrieval search engines. Also, the user has similar to the well-known Yahoo! directory (*cf.* (Labrou & Finin, 1999)) a context that allows for refining or generalizing a query. An example is shown in Section 5.4 in Figure 5.3.

In a nutshell, the upper two levels in the conceptual architecture of SEAL (as shown in Figure 12.1) are implemented as KAON Portal (further details can be found *e.g.* in (Motik et al., 2002; Bozsak et al., 2002)). It generates content objects and provides browsing as well as a query frontend. The lower part of the conceptual architecture, *viz.* the replication of distributed knowledge into the storage facility of the portal, is done by the collaboration of the KAON Syndicator and the DOGMA Server. Please

[8]Protégé Project, see `http://protege.stanford.edu/`

note that only gathered meta data is replicated and not, *e.g.*, documents. The storage consists of (i) a content management system that allows for creation and management of documents (but not annotations), (ii) the RDF management system that stores ontologies and associated annotations, *i.e.* meta data, of the content management system.

The syndication process works as follows: the OntoWeb community members provide meta data on their web sites. The meta data are then syndicated into the OntoWeb portal with the KAON Syndicator and are available through the query and browsing interface of the OntoWeb portal. Since these issues exceed the scope of the work we would like to mention the following links to further information:

- Detailed information about the querying and browsing interface that is provided by our project partner StarLab at the Free University of Brussels can be found in (Spyns et al., 2002b). The underlying DOGMA approach, also provided by StarLab, is described *e.g.* in (Jarrar & Meersman, 2002; Spyns et al., 2002a).
- Further information on how to annotate documents, *e.g.* by using the OntoMat–*Annotizer*, are provided by the S-CREAM methodology for (semi-automatically) annotating documents with meta data (*cf.* (Handschuh et al., 2001; 2002)).
- The idea on collecting meta data from community members originates from the KA2 Community Web Portal, *cf.* (Benjamins & Fensel, 1998; Benjamins et al., 1999; Staab et al., 2000) for further information. They proposed an extension of HTML 4.0 (*cf.* (Raggett et al., 1998)), *viz.* HTML-A (*cf.* (Erdmann et al., 2000), to annotate HTML pages.

The basic content management features, including the workflow component described in the next section, are provided by the CMF framework[9], an extension of the ZOPE web application server[10]. ZOPE and CMF provide the necessary basic infrastructure for the portal, *viz.* a basic content management framework (CMF) that is flexible and easily extensible. They are both available as open source software, therefore we could re-use the existing components and could extend them with our own (ontology-based) modules.

12.2.5 Process Model

OntoWeb as such is an open community. Open communities pose additional constraints since data that is (re)published through the portal could be provided by arbitrary

[9]CMF stands for “Content Management Framework”, see `http://cmf.zope.org/`
[10]ZOPE, see `http://www.zope.org/`

people. In order to guarantee quality of data in such an environment an additional model regulating the publishing process is required, which prevents foreseeable misuses. To support this requirement the established SEAL architecture was extended with a workflow component which regulates the publishing process. In the following we will begin with introducing the concept of a publishing workflow in general. Afterwards we explain how we instantiated this generic component in OntoWeb.

Publishing Workflows

A publishing workflow is the series of interactions that should happen to complete the task of publishing data. Business organizations have many kinds of workflow. Our notion of workflow is centered around tasks. Workflows consist of several tasks and several transitions between these tasks. Additionally workflows have the following characteristics: (i) they might involve several people, (ii) they might take a long time, (iii) they vary significantly in organizations and in the computer applications supporting these organizations respectively, (iv) sometimes information must be kept across states, and last but not least, (v) the communication between people must be supported in order to facilitate decision making. Thus, a workflow component must be customizable. It must support the assignment of tasks to (possibly multiple) individual users. In our architecture these users are grouped into roles and tasks are represented within a workflow as a set of transitions which cause state changes. Each object in the system is assigned a state, which corresponds to the current position within the workflow and can be used to determine the possible transitions that can validly be applied to the object. This state is persistent supporting the second characteristic mentioned above. Due to the individuality of workflows within organizations and applications we propose a generic component that supports the creation and customization of several workflows.

In fact, each concept in the ontology, which is used to capture structured data within a portal, can be assigned a different workflow with different states, transitions and task assignments. As mentioned above, sometimes data is required to be kept across states. For example, envision the process of passing bills in legislature, a bill might be allowed to be revised and resubmitted once it is vetoed, but only if it has been vetoed once. If it is vetoed a second time, it is rejected forever. To model this behavior, the state machine underlying our workflow model needs to keep information that “remembers” the past veto. Thus, variables are attached to objects and used to provide persistent information that transcends states. Within our approach variables also serve the purpose of establishing a simple form of communication between the involved parties. Thus, each transition can attach comments to support the decision made by future actors. Also meta data like the time and initiator of a transition is kept within the system.

Workflow in the OntoWeb Portal

Figure 12.3 depicts the default publishing workflow within the OntoWeb portal. In the portal there exist three different types of user roles: *user*, *reviewer* and *manager*. Typically, all portal users have assigned the role *user*, additionally they might be *reviewer* or *manager*. Furthermore, there exist three states: *private*, *pending*, and *published*.

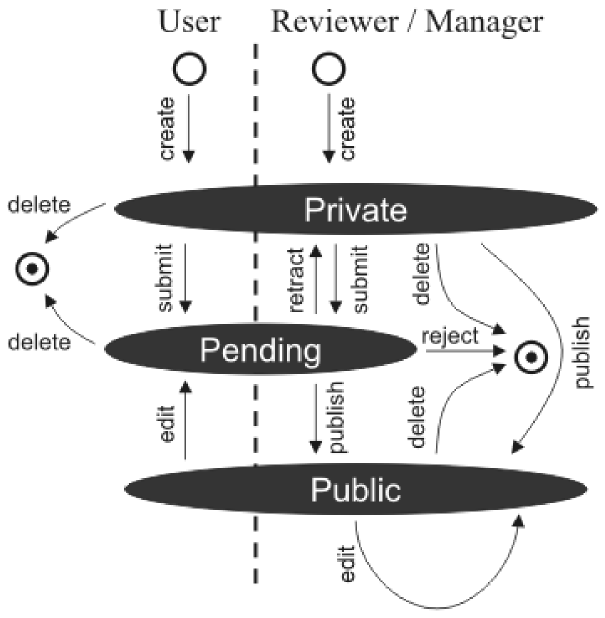


Figure 12.3: Publishing workflow in the OntoWeb portal (i)

In the *private* state the respective object is only visible to a *user* himself, the pending state makes it additionally visible to *reviewers*. In the *published* state, a given object is visible to all (possibly anonymous) *users* of the portal. If a *user* creates a new object[11] the object is in *private* state. If a *user* has additionally either a *reviewer* or a *manager* role the *published* state is immediately available through the publish transition. For "normal" *users* such a transition is not available, instead the object can only be sent for a review leading to the *pending* state. In the *pending* state either *managers* or *reviewers* can do the transition to the *published* state (by applying the transition *publish*) or retract the object leading back to the *private* state.

The *reject* transition deletes the object completely. When an object is in the *private* state, only the *user* who created it and users with *manager* roles can view and change

[11]This is currently only true within the portal, the content syndicated from other OntoWeb member web sites is trusted. We assume that this kind of data already went through some kind of review.

it. Once an object is in *published* state the modification by the *user* who created it resets the object into *pending* state, thus the modification must be reviewed again. This does not apply to modifications by site *managers*.

We will illustrate the publishing process by an example from the OntoWeb portal itself. Figure 12.4 shows a compilation of a series of relevant parts of screenshots that were captured during a typical review process. The numbers shown in the figure refer to the following actions:

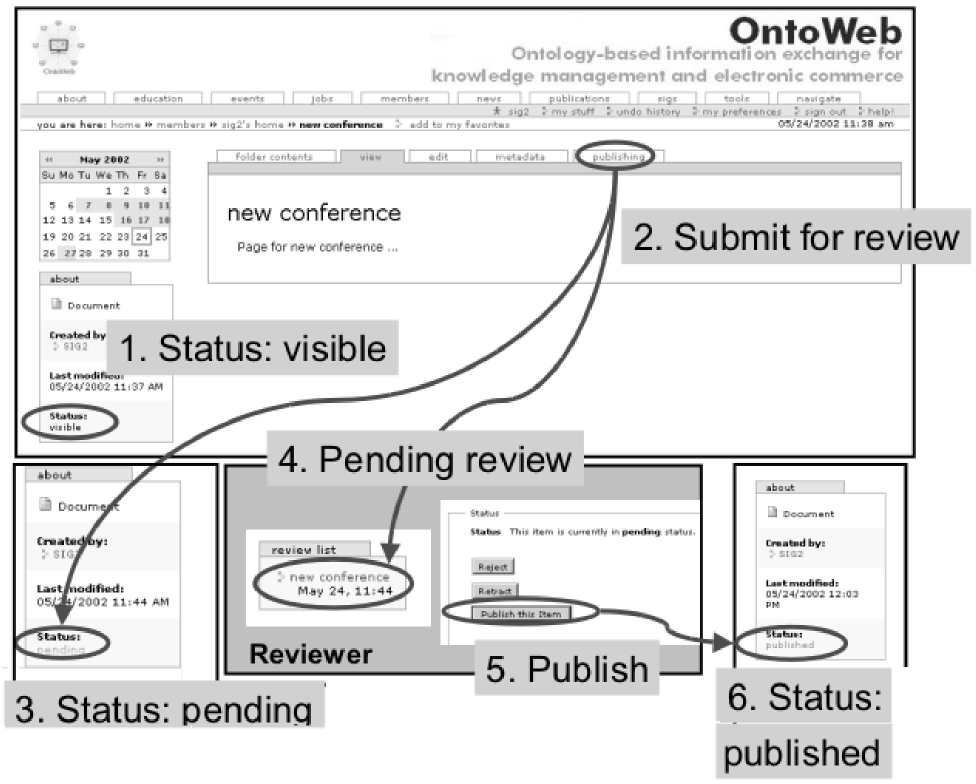


Figure 12.4: Publishing workflow in the OntoWeb portal (ii)

1. After a *user* entered a new conference, this newly created content object has the state *visible*. It is now visible for the *user*, but not for other *users*.

2. The *user* submits the object for review.

3. The object has now the state *pending*.

4. A *reviewer*[12] is assigned to review the *pending* object.

[12]Currently all reviewers get reviewing requests, in the future it is planned to assign reviews according

5. After checking the content the *reviewer* publishes the object.
6. The object is now in the state *published* and therefore publicly available.

However, the work on the OntoWeb portal is an ongoing effort, we will now sketch the future directions for the development of the OntoWeb portal.

12.2.6 Future Directions

It is important to mention that in our current SEAL architecture (and implementation) we mainly apply static information integration building on a warehousing approach. Means for dynamic information integration are currently approached for Peer-2-Peer (P2P) networks, *e.g.* in the PADLR project[13]. P2P applications for searching and exchanging information over the Web have become increasingly popular. The "Edutella"[14] approach (*cf.* (Nejdl et al., 2002b; 2002a)) builds upon the RDF meta data standard aiming to provide an RDF-based meta data infrastructure for P2P applications, building on the recently announced JXTA framework[15].

Currently there exist two other ontology-based portals developed by members of OntoWeb, *viz.* the Roadmap Portal[16] and the Edu Portal[17], each portal having its own ontology as a backbone. Currently the next task for the AIFB and related partners is to integrate all three portals. There exist three different layers for a possible integration approach, each requiring different steps.

1. **Meta data integration:**
   a) Align the three ontologies to meet different requirements[18].
   b) Publish meta data according to aligned ontology in all three portals.

to personal profiles of the reviewers.

[13]PADLR stands for "Personalized Access to Distributed Learning Repositories", see `http://www.learninglab.de/workspace/padlr/`

[14]Edutella project, see `http://edutella.jxta.org`

[15]JXTA *"is short for Juxtapose, as in side by side. It is a recognition that peer to peer is juxtapose to client server or Web based computing – what is considered today's traditional computing model"*, *cf.* `http://www.jxta.org/`

[16]Roadmap Portal, provided by the Universidad Politecnica de Madrid (UPM), see `http://babage.dia.fi.upm.es/ontoweb/wp1/OntoRoadMap/index.html`

[17]Edu Portal, provided by the Knowledge Media Institute at the Open University and the Queen Mary University of London, see `http://qmir.dcs.qmul.ac.uk/ontoweb/`

[18]An approach for mapping ontologies is *e.g.* presented in (Maedche et al., 2002a). The OntoMap plugin presented in Section 8.3 already provides an initial implementation for generating mappings. However, all related partners in OntoWeb agreed to start with one single consolidated ontology and to extend the scenario in the future. Main reason for this is to get up and running integrated portals as soon as possible.

   c) Syndicate meta data from Edu/Roadmap Portals into the Semantic Portal.
   d) **Note:** This means that contents of the Edu/Roadmap Portals are exported to RDF (according to the OntoWeb ontology) and syndicated into the Semantic Portal. This would actually reflect the current architecture of OntoWeb.

2. **Layout alignment:**
   a) Use the layout from the OntoWeb Semantic Portal, *i.e.* use Edu/Roadmap Portal technology, but align layouts to the Semantic Portal to create a uniform look & feel
   b) Unify website layout to the one from the Semantic Portal by using the style files of the Semantic Portal.
   c) Add additional navigational tabs in the interface of the Semantic Portal for Edu/Roadmap Portals that link to the other – still separate – portals.
   d) **Note:** This means that the existing portals remain separate, but a uniform look is provided and the top level navigational structure is the same for all three portals.

3. **Technology integration:**
   a) The content and functionalities of the Edu/Roadmap Portals are transferred into the Semantic Portal, *i.e.* the Semantic Portal is extended by needed functionalities and covers afterwards also the content and functionalities of the Edu/Roadmap Portals.
   b) **Note:** This requires the complete re-implementation of the Edu/Roadmap Portals.

Discussions between all related partners led to the consensus to head for the first option. This consensus is mainly based on the following reasons:

- The solution fits perfectly in the overall OntoWeb portal architecture. The two additional portals are treated as any other web site of OntoWeb members that contributes meta data to the OntoWeb Semantic Portal via syndication.
- The Semantic Web is likely to be highly distributed, the three portals serve as show cases for Semantic Web technologies and should reflect that.

12.3 Evaluation Efforts

In this section we present work on evaluation of ontology based tools. This work was performed as part of the work of the "Special Interest Group on Tools (SIG4)" of the OntoWeb thematic network and it presents the state-of-the-art on evaluation of such tools. We introduce the aims of the EON workshop in Section 12.3.1 and illustrate the experiment that was carried out as part of it in the Section 12.3.2 and our contribution to the experiment in Section 12.3.3. After presenting the results of the workshop in Section 12.3.4, we finally sketch future directions for evaluation research on ontology based tools in Section 12.3.5.

12.3.1 Workshop Description

In the "Evaluation of Ontology-based Tools (EON 2002)"[19] workshop the intention was to bring together researchers and practitioners from the quickly developing research areas "ontologies" and "Semantic Web". Currently the semantic web attracts researchers from all around the world. Numerous tools and applications of Semantic Web technologies are already available and the number is growing fast. However, deploying large scale ontology solutions typically involves several separate tasks and requires applying multiple tools. Therefore pragmatic issues such as interoperability are key if industry is to be encouraged to take up ontology technology rapidly.

The main aim of this workshop was therefore to encourage and stimulate discussions about the evaluation of ontology-based tools. For the future this effort might lead to benchmarks and certifications.

The workshop was divided in two parts: (i) presentations of accepted papers and (ii) discussion of the "EON 2002 Experiment". The experiment was initiated during the OntoWeb–3 meeting[20] by the participants of the Special Interest Group on Tools (SIG3). The general question was how to evaluate ontology related technologies. To brake down this rather complex task into a pragmatic one, the group decided to focus on ontology engineering environments (OEE) as a starting point. These tools are rather common and widely used by the Semantic Web Community and some of the participating members were even tool provider themselves.

It was not intended to provide "yet another ranking of tools", *e.g.* XML.com recently

[19]EON 2002, held on 30th September 2002 in Siguenza, Spain (*cf.* (Sure & Angele, 2002)). The workshop was co-located with the conference EKAW 2002 (*cf.* (Gómez-Pérez & Benjamins, 2002)). Further information, including the proceedings and the experiment contributions, can be found at `http://km.aifb.uni-karlsruhe.de/eon2002/`.

[20]The OntoWeb–3 meeting was held from 12–14th June 2002 in Sardinia, *cf.* `http://www.ontoweb.org/` for further information.

provided a list of currently available "ontology editors"[21]. Interestingly enough, many of the tools were unknown even to participants of the OntoWeb Special Interest Group on Tools (SIG3). Tools were listed with their sets of features, but simply listing those features does unreveal how to use them for modelling! Thus, in the experiment it was rather intended to stimulate discussions about design rationales behind the tools that point to directions for future evaluation series. Therefore, an important aspect of the experiment was to keep it as open as possible without putting constraints on the participants to get unbiased pictures of their modelling habits.

For the future the experiment should provide a first basis for "ontology developers guidelines" to support their decision which tool to use for modelling ontologies for specific purposes.

The experiment was based on a domain description, *viz.* the travelling domain, for which an ontology should be modelled by using a preferred OEE. Submissions to the experiment should include (i) a brief description about design decisions and (ii) the resulting ontology itself. The description should answer the following items with respect to the used OEE:

- What modeling decisions need to be considered during the design?
- What limitations occur?
- ... and why?
- What problems arise due to using different representation languages for export?
- What are the lessons learned from modelling this experiment?

The generated ontologies should be exported into a common representation language. However, most OEEs were designed having specific design rationales from representation formalisms in mind. Therefore they typically have a strong support for their "home language". To make the results more comparable people were encouraged to provide not only an "home language" export, but also an RDF(S) export.

The next Sections 12.3.2 and 12.3.3 contain the domain description for the travelling domain and a description on how the domain was modelled with OntoEdit. We then conclude by presenting some overall results of the workshop in Section 12.3.4.

12.3.2 Travelling Domain Description

This description of the domain was initially taken for the experiment from (Gómez-Pérez et al., 2002a).

[21]XML.com, *cf.* http://www.xml.com/pub/a/2002/11/06/ontologies.html

"Let's consider that we are in charge of developing an application for our travel agent in New York, and that we have decided to make use of an ontology to represent explicitly the knowledge that will be used by it. We will focus to travelling and lodging, but leisure time, cultural events, tours, etc., will be considered in further stages of our ontology.

We know that when a client makes a trip, he chooses: transport and accommodation.

Hence, we start by determining the means of transport that are currently available for a travel agency. We will have in our ontology the following ones: planes, trains, cars, ferries, motorbikes and ships. There are no other kinds of transport. From all of them, the travel agency is specially interested in flights, as it is the means of transport mostly used by its customers. In fact, customers are usually interested in the kind of planes that they will fly on: Is it a Boeing, or is it an Airbus? Furthermore, they are even interested in the specific model of the plane in which they will fly (a Boeing 717 or a Boeing 777). We know that each model of transport belongs only to one kind of transportation (e.g., it's either a plane, or a bus, or a car, etc.).

For each flight, the agency knows: the arrival date, the departure date, the arrival city, the departure city, the arrival airport, the departure airport, the prices on first class, business class and economy class, the departure time and arrival time. Time and date will be considered as absolute date.

As for the destinations of customers' travels, they are diverse. Some customers ask for trips to the Statue of Liberty in New York; other ask for trips to Washington, San Francisco, Seattle. There are customers interested in visiting Europe: the most common destinations are London, Paris (either the city or Disneyland Paris) and Madrid. Others are interested in more places, such as Cairo (Egypt). We know that the client can use the following transport to move inside the city: underground, city buses, taxis, and rental cars.

Concerning hotels, the agency recommends in all the cities: hotels, and Bed and Breakfasts. Hotels rank from 1 star hotels to 5 star hotels and each hotel belongs to one of these five categories. For all of them, the agency knows their facilities: address, telephone number, URL, capacity, number of rooms, available rooms, descriptions, dogs allowed, distance to the beach, distance to skiing, etc. The agency also knows the facilities of the rooms: number of beds, rates, TV available, Internet connection, etc.

Once we have defined what are the main elements in our domain, we can go further and try to represent some common sense constraints and deductions that can be performed with them. For instance, we know that it is not *possible to go from America to Europe by train, car, bike nor motorbike. Having this information in our system will avoid it to search for possible itineraries using these means of transport when a customer wants to travel to Europe. Another example of this kind of constraint may be related to the distance between the origin and destination of our trip and the available means of transport. If distance between two cities is between 400 and 800 miles, and there is no airport close to one of them, the customer will prefer going by car or by train. The customer also prefer to go by car or train if he hates travel by plane. Distances can be either in km or miles.*

Finally, we want to represent knowledge about a concrete trip. John is travelling from Madrid to NY on April 5th, 2002 to see the Statue of Liberty and continuing on to Washington on April 11th. He plans to return to Madrid on April 15th. He has selected two hotels belonging to the Holiday Inn chain in New York and Washington.

http://www.boeing.com/commercial/717/717technical.html provides Boeing 717 technical description."

12.3.3 Engineering the Model with OntoEdit

For engineering the travelling domain model with OntoEdit (*cf.* (Sure, 2002c)) we performed the two Knowledge Meta Process steps "Kickoff" and "Refinement" of the On-To-Knowledge Methodology.

Kickoff

In the first step, a semi-formal description of the ontology is created by sketching the most relevant elements of the domain. The early stages of ontology development are often driven by brainstorming like knowledge acquisition sessions. In other projects (*cf.*, *e.g.*, Chapter 11) we made good experiences with creating mind maps™ as a first draft of relevant elements for a domain. Especially domain experts who were not familiar with modeling preferred using a mind mapping tool instead of directly modeling with an ontology editor. Figure 12.5 shows the mind map™ created from the natural language description of the domain. We rely on a commercial tool for the creation of electronically mind maps™, the MindManager 2002 Business Edition[22].

When collaborating with domain experts the time needed for knowledge acquisition is essential, especially in industrial environments. The advantage of using mind maps™ is (i) the quick generation of a graphical representation of relevant domain elements

[22]MindManager 2002 Business Edition, http://www.mindjet.com/

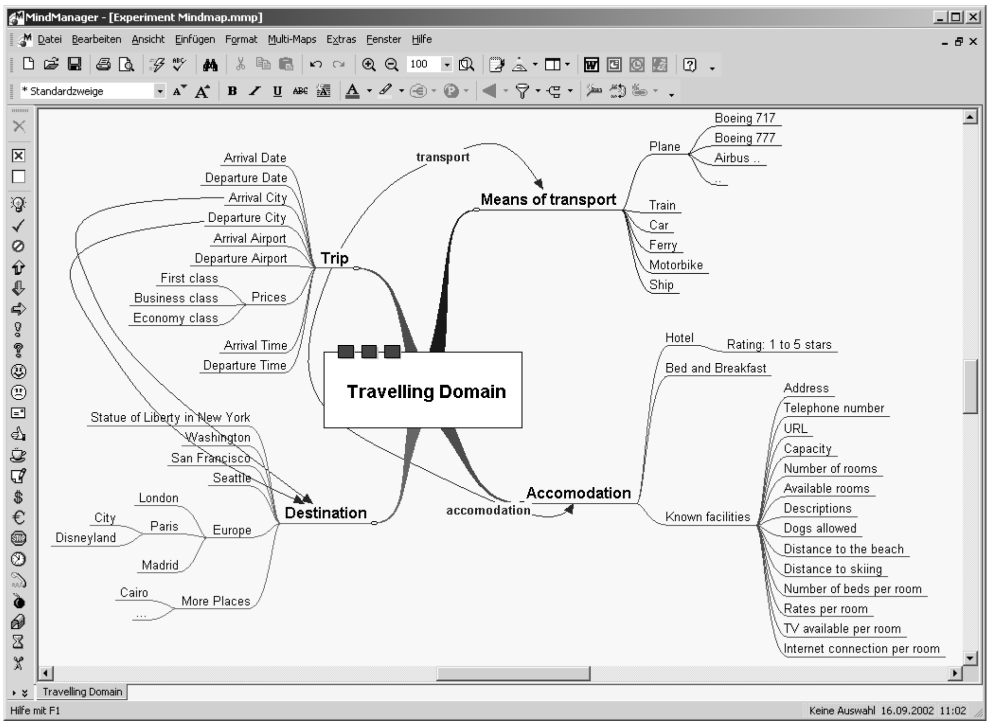


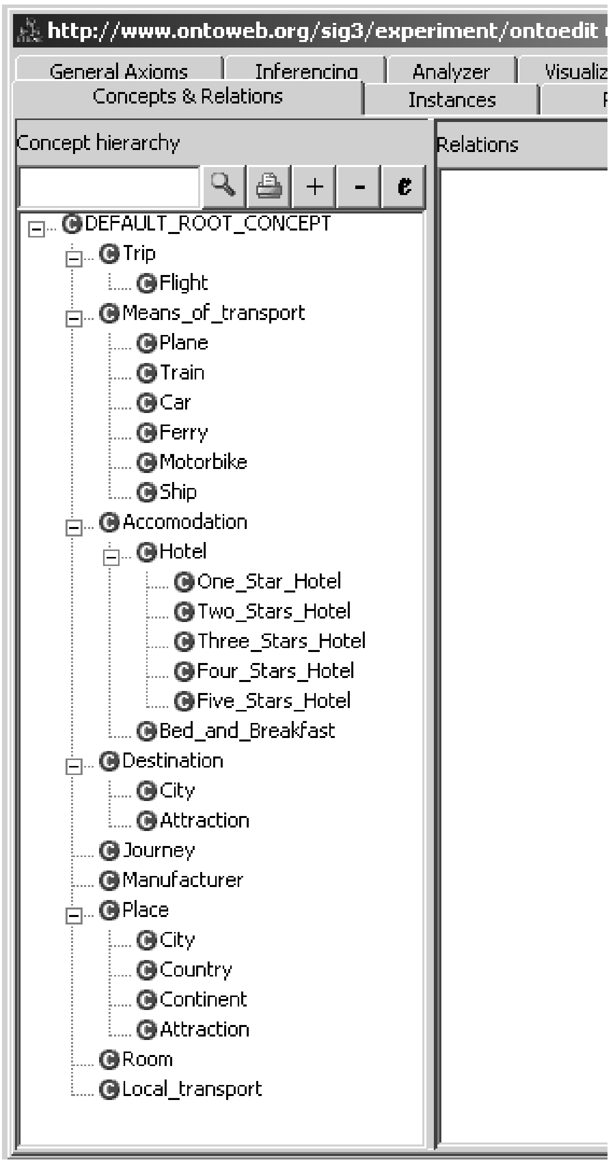
Figure 12.5: MindMap of the travelling domain

(ii) by using an intuitive and rather well-known tool. The creation of this mind map™ took less than 20 minutes.

However, when it comes to terms of a formal model of the domain, this representation is no longer suitable. This representation does not clearly distinguish between the notions of concepts, relations *etc.* The only semantics for connections (branches or directed edges) in a mind map™ is that these elements are "associatively linked". Closer related elements are typically marked with the same colors. Typically a mind map™ represents the key concepts and their relationships and to formalize it into an ontology, the ontology engineer has now to decide which elements are concepts, how is the hierarchical "is–a" structure and which elements are other named relationships. In some cases one might find prototypical instances, but constraints like the ones given at the end of the domain description are typically not found in mind maps™.

Refinement

The Mind2Onto framework supports the transfer of the mind map™ into OntoEdit (*cf.* Section 7.2). To formalize the mind map™ we followed the steps (i) creation of an "is–a" hierarchy of concepts, (ii) adding attributes of concepts and relationships between concepts other than "is–a", (iii) including prototypical instances and (iv) adding axioms that represent constraints and common sense deductions. At several points we needed to introduce further concepts, that were not obvious at first hand from the domain description but necessary to build a complete model. *E.g.* we introduced a concept Journey to combine several trips, some others are mentioned in the text below. This reflects the fact that the mind map™ is typically covering only the most relevant elements of a domain, but is not intended to represent more complex relationships in a formally consistent way.

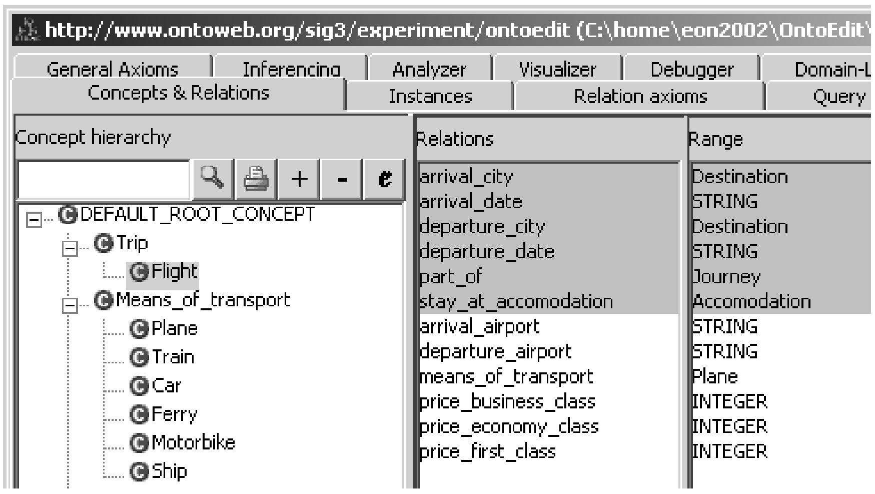


```
DEFAULT_ROOT_CONCEPT
    Trip
        Flight
    Means_of_transport
        Plane
        Train
        Car
        Ferry
        Motorbike
        Ship
    Accomodation
        Hotel
            One_Star_Hotel
            Two_Stars_Hotel
            Three_Stars_Hotel
            Four_Stars_Hotel
            Five_Stars_Hotel
        Bed_and_Breakfast
    Destination
        City
        Attraction
    Journey
    Manufacturer
    Place
        City
        Country
        Continent
        Attraction
    Room
    Local_transport
```

Figure 12.6: Concept hierarchy

Figure 12.6 shows the resulting concept hierarchy. When defining a Trip we made the following assumption to narrow down multiple possible interpretations in the description: Not only for flights, but for every Trip the arrival and departure date, arrival and departure city is known.

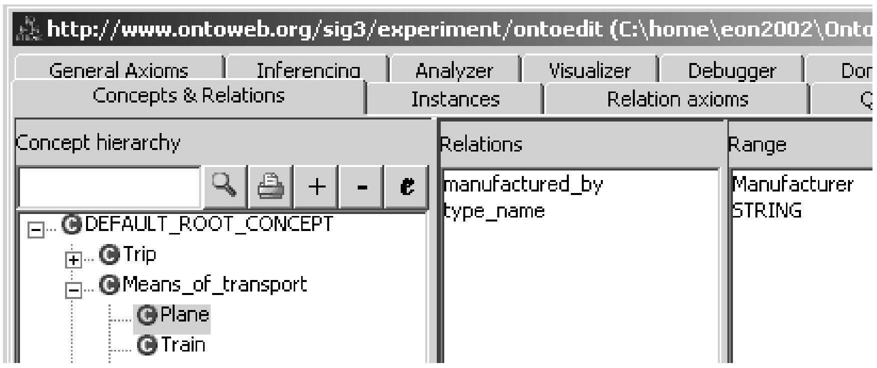
A Flight is a specialization of Trip for which in addition the arrival and departure airport and the prices for first, business and economy class are known. The grey shaded relationships shown in Figure 12.7 illustrate the inherited relationships for the selected concept Flight, *i.e.* the domain of them is Trip. The relationships without shading have as a domain Flight itself.



Relations	Range
arrival_city	Destination
arrival_date	STRING
departure_city	Destination
departure_date	STRING
part_of	Journey
stay_at_accomodation	Accomodation
arrival_airport	STRING
departure_airport	STRING
means_of_transport	Plane
price_business_class	INTEGER
price_economy_class	INTEGER
price_first_class	INTEGER

Figure 12.7: Relationships of Flight

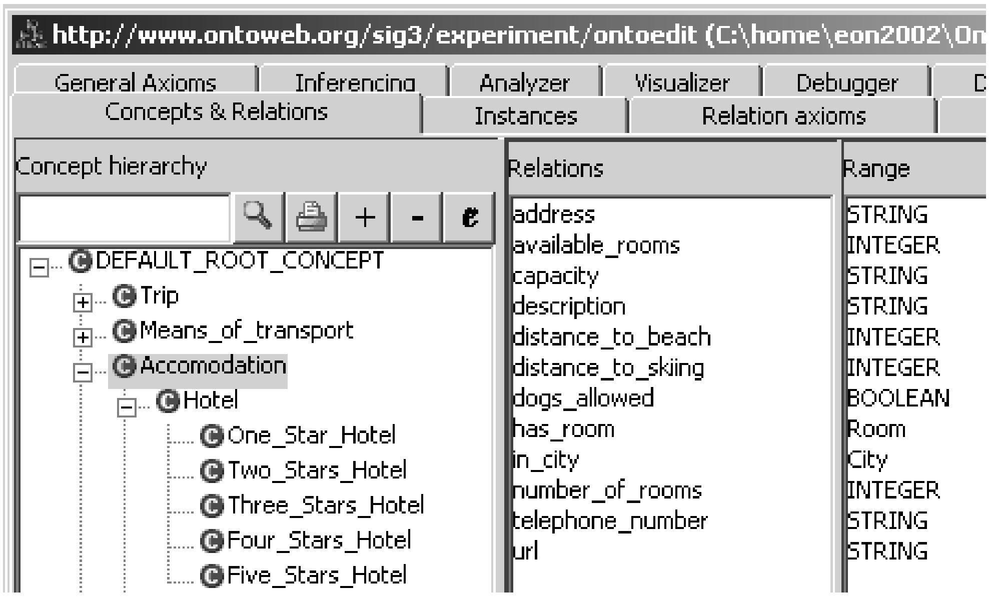
We made some simple assumptions for defining ranges of the relationships: *e.g.* the dates are coded as STRING which directly points to the XML Schema definition for strings (`http://www.w3.org/2001/XMLSchema#String`) – the same holds *e.g.* for prices/INTEGER. The relationship MEANS OF TRANSPORT is firstly defined for Trip with the range Means of transport. We then refined it for Flight by specializing the range to Plane that is a subconcept of Means of transport (*cf.* Figure 12.8).



Relations	Range
manufactured_by	Manufacturer
type_name	STRING

Figure 12.8: Relationships for Plane

Accomodation (*cf.* Figure 12.9) has the subconcepts Hotel and Bed and Breakfast. Beside relationships for the facilities mentioned in the description (ADDRESS, AVAILABLE ROOMS *etc.* ..) one can see relationships to Room and City.



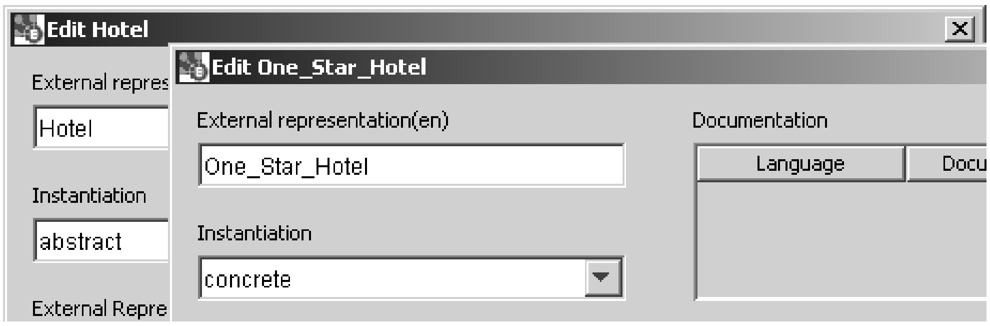
Concept hierarchy

```
DEFAULT_ROOT_CONCEPT
    Trip
    Means_of_transport
    Accomodation
        Hotel
            One_Star_Hotel
            Two_Stars_Hotel
            Three_Stars_Hotel
            Four_Stars_Hotel
            Five_Stars_Hotel
```

Relations	Range
address	STRING
available_rooms	INTEGER
capacity	STRING
description	STRING
distance_to_beach	INTEGER
distance_to_skiing	INTEGER
dogs_allowed	BOOLEAN
has_room	Room
in_city	City
number_of_rooms	INTEGER
telephone_number	STRING
url	STRING

Figure 12.9: Relationships of Accomodation

To model the star ranking schema for hotels we added further specializations of Hotel, *e.g.* One-Star-Hotel (see later in the subsection about instances how we model a particular hotel as an instance). To model that each hotel belongs to one star category, we defined Hotel as an "abstract" concept, *i.e.* there are no instances of this concept allowed, and all subconcepts like One-Star-Hotel as "concrete" concepts (*cf.* Figure 12.10). Same holds *e.g.* for Means of transport and its subconcepts.



External repres: Hotel

Instantiation: abstract

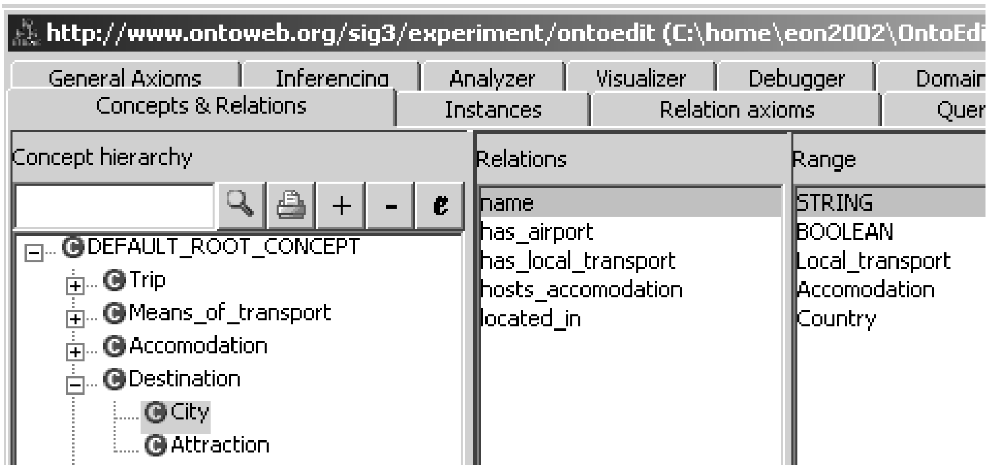
External representation(en): One_Star_Hotel

Instantiation: concrete

Figure 12.10: "Abstract" *vs.* "concrete" concepts

We introduced Destination as a superconcept of City (*cf.* Figure 12.11). For further axioms on top we also included Attraction, Country and Continent. The concepts are related via the LOCATED IN relationship, *e.g.* an Attraction IS LOCATED IN a City, a City IS LOCATED IN a Country and a Country IS LOCATED IN a Continent. As shown later this relationship is transitive. City and Attraction are also subconcepts of Destination, *i.e.* they are multiply inherited. Alternatively one could consider to add Destination as a subconcept of Place, too. In the current scenario that would have no effect. Modelling it this way seemed more intuitive to us.

Last but not least, a Journey has potentially many parts, *i.e.* it can be related via HAS PART to many instances of Trip that belong to this Journey. For completeness we included also the inverse relationship PART OF for Trip with the range Journey and defined these two relationships as invers (see later paragraph on axioms).

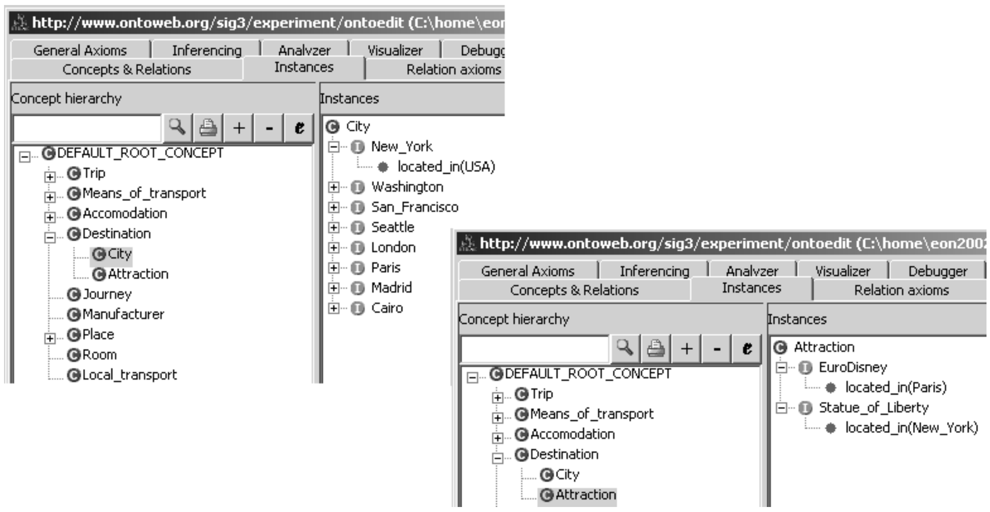


Relations	Range
name	STRING
has_airport	BOOLEAN
has_local_transport	Local_transport
hosts_accomodation	Accomodation
located_in	Country

Figure 12.11: Relationships of City

We modelled several instances, *e.g.* shown in Figures 12.12, 12.13 and 12.14[23]. Part of them were given in the first part of the description (*e.g.* the cities `New York`, `Washington` *etc.* and the attractions `Statue of Liberty` and `EuroDisney`), others were given in the last section with an example journey for John (*cf.* Figure 12.14). As shown, John makes two flights (from Madrid to NY and from Washington to Madrid) and one trip with a motorcycle (from NY to Washington).

[23]It is noteworthy that the instances shown in Figure 12.13 are not directly instances of Means of Transport, but rather of it's subconcepts. However, OntoEdit shows in this instance view for a selected concept all instances from the concept itself and its subconcepts.



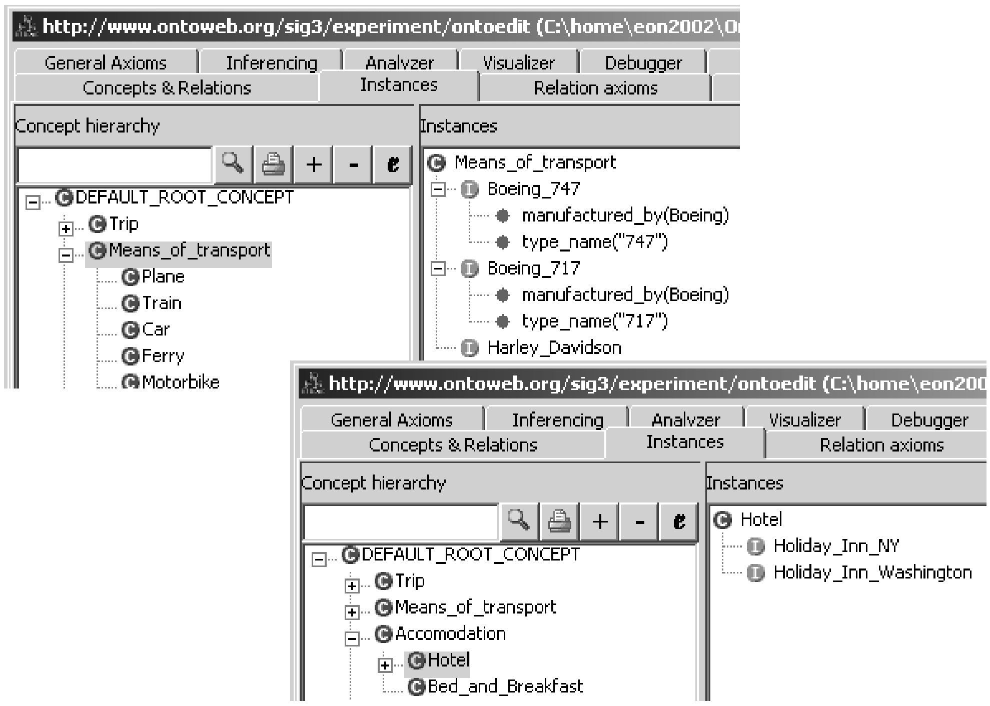
```
DEFAULT_ROOT_CONCEPT
  Trip
  Means_of_transport
  Accomodation
  Destination
    City
    Attraction
  Journey
  Manufacturer
  Place
  Room
  Local_transport
```

```
City
  New_York
    located_in(USA)
  Washington
  San_Francisco
  Seattle
  London
  Paris
  Madrid
  Cairo
```

```
DEFAULT_ROOT_CONCEPT
  Trip
  Means_of_transport
  Accomodation
  Destination
    City
    Attraction
```

```
Attraction
  EuroDisney
    located_in(Paris)
  Statue_of_Liberty
    located_in(New_York)
```

Figure 12.12: Instances of City and Attraction



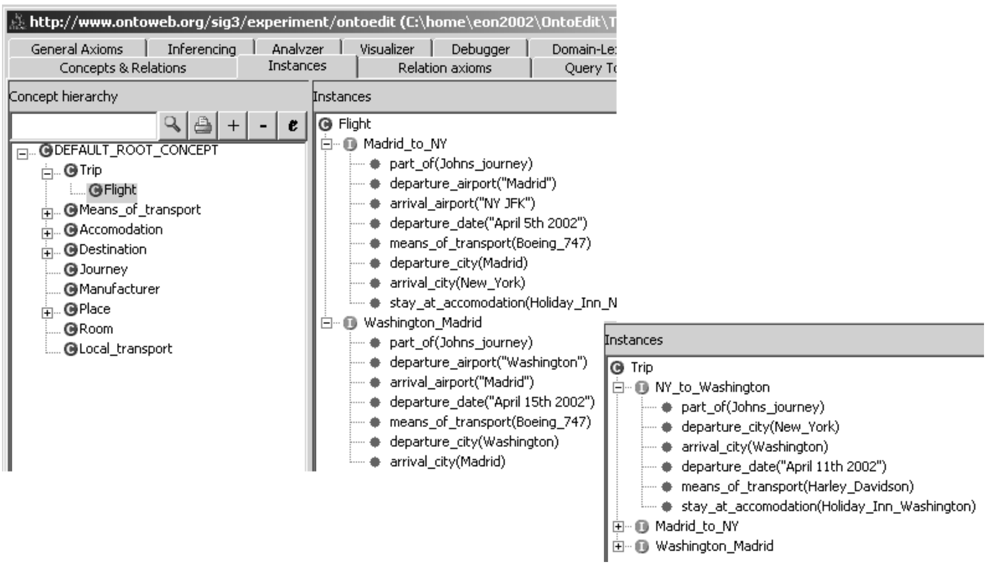
```
DEFAULT_ROOT_CONCEPT
  Trip
  Means_of_transport
    Plane
    Train
    Car
    Ferry
    Motorbike
```

```
Means_of_transport
  Boeing_747
    manufactured_by(Boeing)
    type_name("747")
  Boeing_717
    manufactured_by(Boeing)
    type_name("717")
  Harley_Davidson
```

```
DEFAULT_ROOT_CONCEPT
  Trip
  Means_of_transport
  Accomodation
    Hotel
    Bed_and_Breakfast
```

```
Hotel
  Holiday_Inn_NY
  Holiday_Inn_Washington
```

Figure 12.13: Instances of Means of transport and Three Stars Hotel



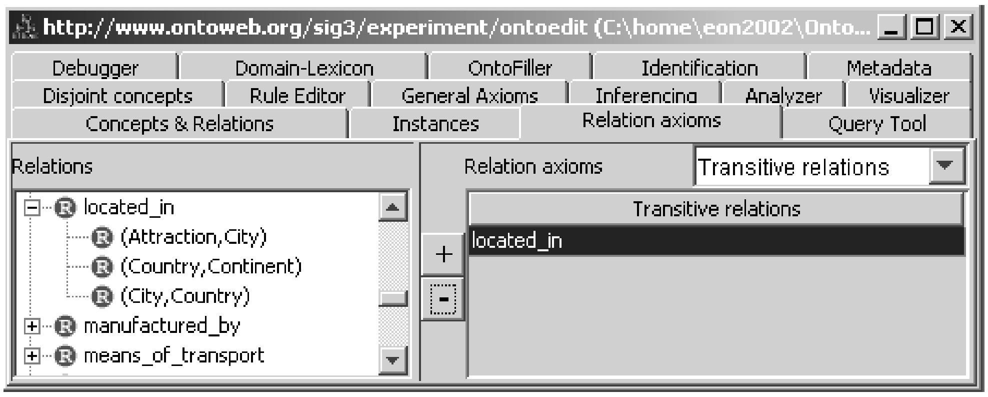
```
DEFAULT_ROOT_CONCEPT
  Trip
    Flight
  Means_of_transport
  Accomodation
  Destination
  Journey
  Manufacturer
  Place
  Room
  Local_transport
```

```
Flight
  Madrid_to_NY
    part_of(Johns_journey)
    departure_airport("Madrid")
    arrival_airport("NY JFK")
    departure_date("April 5th 2002")
    means_of_transport(Boeing_747)
    departure_city(Madrid)
    arrival_city(New_York)
    stay_at_accomodation(Holiday_Inn_N
  Washington_Madrid
    part_of(Johns_journey)
    departure_airport("Washington")
    arrival_airport("Madrid")
    departure_date("April 15th 2002")
    means_of_transport(Boeing_747)
    departure_city(Washington)
    arrival_city(Madrid)
```

```
Trip
  NY_to_Washington
    part_of(Johns_journey)
    departure_city(New_York)
    arrival_city(Washington)
    departure_date("April 11th 2002")
    means_of_transport(Harley_Davidson)
    stay_at_accomodation(Holiday_Inn_Washington)
  Madrid_to_NY
  Washington_Madrid
```

Figure 12.14: Instances of Flight and Trip

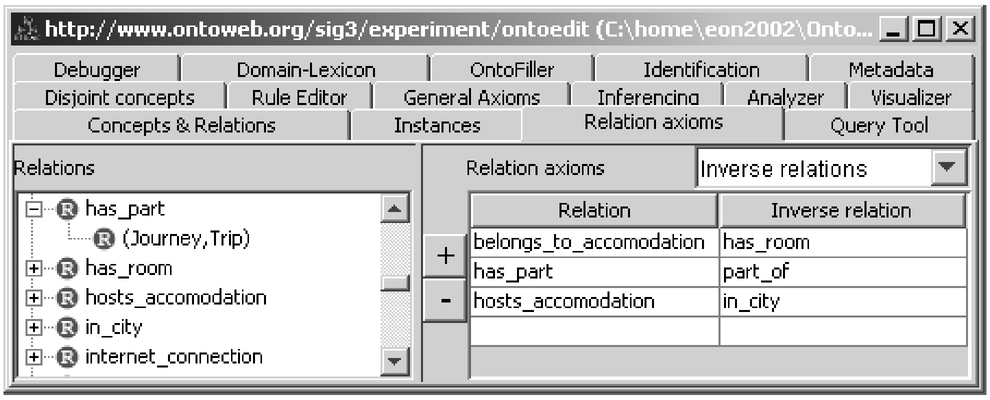
On top we defined several axioms: LOCATED IN is transitive (*cf.* Figure 12.15), *e.g.* HAS PART is inverse to PART OF (*cf.* Figure 12.16), the subconcepts of Hotel are pairwise disjoint (*cf.* Figure 12.17).



```
located_in
  (Attraction,City)
  (Country,Continent)
  (City,Country)
manufactured_by
means_of_transport
```

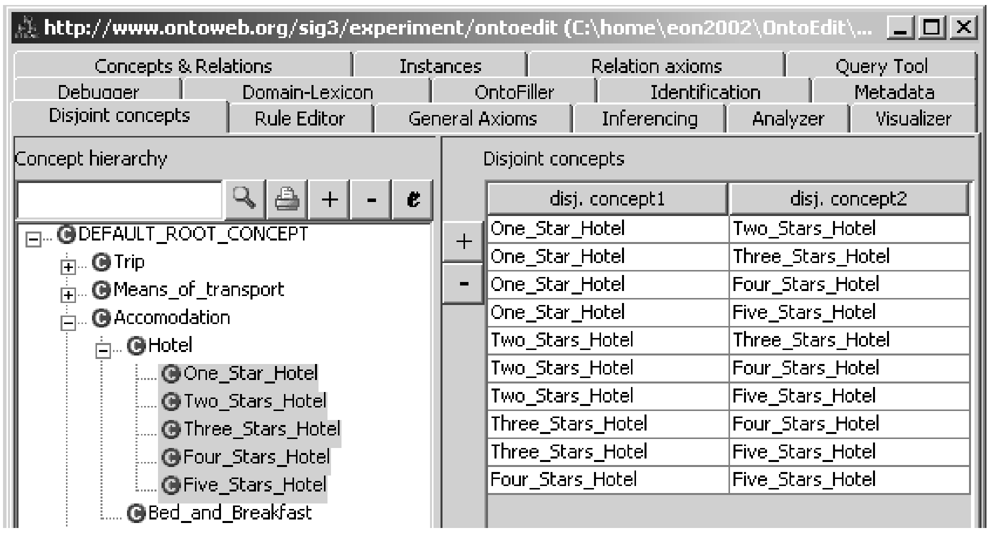
Transitive relations
located_in

Figure 12.15: Transitive relationships



Relation	Inverse relation
belongs_to_accomodation	has_room
has_part	part_of
hosts_accomodation	in_city

Figure 12.16: Inverse relationships



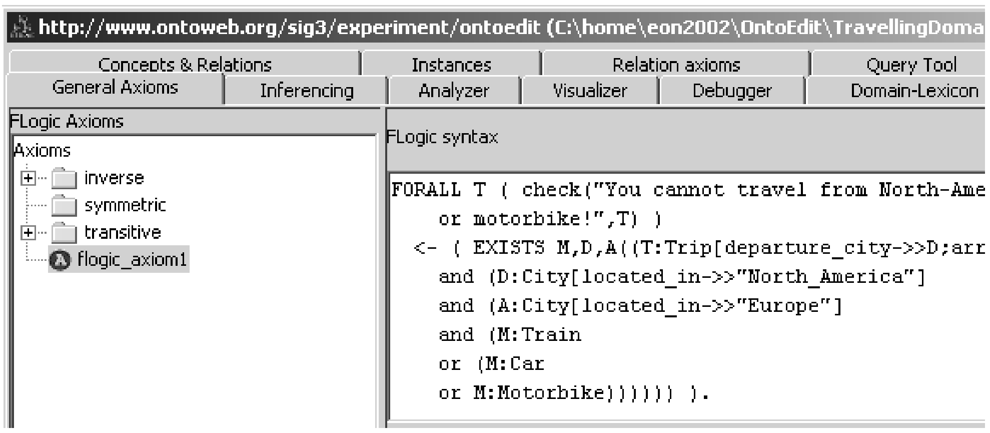
disj. concept1	disj. concept2
One_Star_Hotel	Two_Stars_Hotel
One_Star_Hotel	Three_Stars_Hotel
One_Star_Hotel	Four_Stars_Hotel
One_Star_Hotel	Five_Stars_Hotel
Two_Stars_Hotel	Three_Stars_Hotel
Two_Stars_Hotel	Four_Stars_Hotel
Two_Stars_Hotel	Five_Stars_Hotel
Three_Stars_Hotel	Four_Stars_Hotel
Three_Stars_Hotel	Five_Stars_Hotel
Four_Stars_Hotel	Five_Stars_Hotel

Figure 12.17: Disjoint concepts

A more complex axiom is given in the description by *"it is not possible to go from America to Europe by train, car, bike or motorbike"* (without restricting the generality we excluded bike because it was not given in the previous section for means of transport). We defined a general axiom in F-Logic that can be used to check whether this constraint holds for all given instances (see also Figure 12.18):

```
FORALL T check("You cannot travel from North-America to Europe
               by train, car or motorbike!",T)
   <- EXISTS M,D,A
   T:Trip[departure_city->>D;arrival_city->>A;
         means_of_transport->>M]
   AND D:City[located_in->>"North_America"]
   AND A:City[located_in->>"Europe"]
   AND (M:Train OR M:Car OR M:Motorbike).
```

Other given constraints can be formalized similar to this (*cf.*, *e.g.*, Section 9.2 for more examples). To perform a check we simply query for all values of the 2-ary predicate `check`.

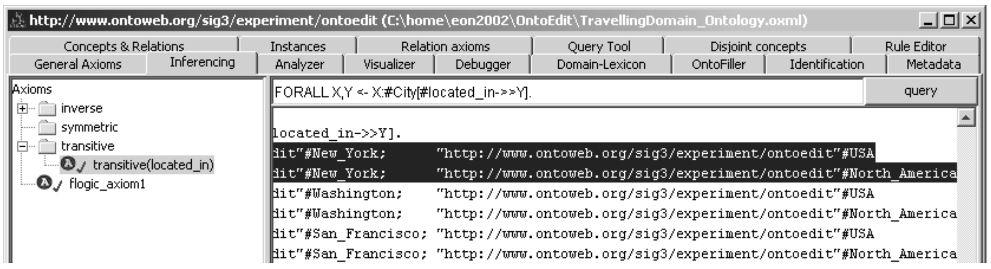


```
FORALL T ( check("You cannot travel from North-Ame
    or motorbike!",T) )
  <- ( EXISTS M,D,A((T:Trip[departure_city->>D;arr
    and (D:City[located_in->>"North_America"]
    and (A:City[located_in->>"Europe"]
    and (M:Train
    or (M:Car
    or M:Motorbike)))))) ).
```

Figure 12.18: General axioms

OntoEdit can be connected to the inference engine Ontobroker (*cf.* Section 6.6). We are thereby able to perform queries for concepts, relationships, instances *etc. E.g.* we can ask for all cities and where they are located in. If we enable the axiom for transitivity of the relationship LOCATED IN (like shown in Figure 12.19) we receive as an answer to that query that *e.g.* `New York` IS LOCATED IN `USA` (an instance of the concept Country) as well as the fact the `New York` IS LOCATED IN `North America` (an instance of the concept Continent).

Disabling the axiom for transitivity of the relationship LOCATED IN (like shown in

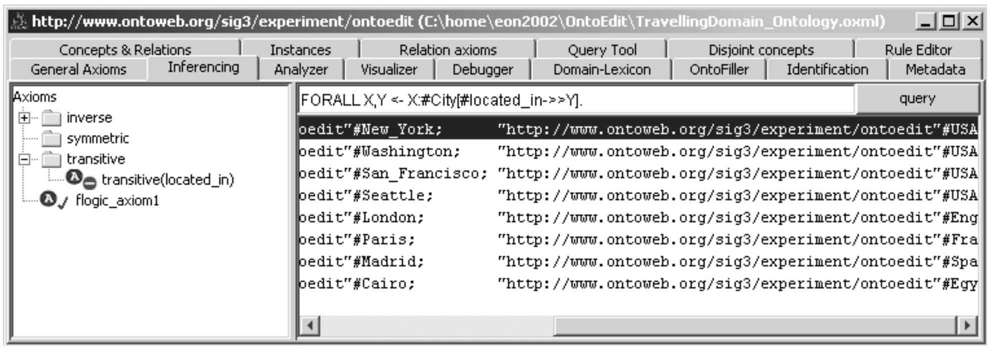


```
FORALL X,Y <- X:#City[#located_in->>Y].
```

```
located_in->>Y].
dit"#New_York;      "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
dit"#New_York;      "http://www.ontoweb.org/sig3/experiment/ontoedit"#North_America
dit"#Washington;    "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
dit"#Washington;    "http://www.ontoweb.org/sig3/experiment/ontoedit"#North_America
dit"#San_Francisco; "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
dit"#San_Francisco; "http://www.ontoweb.org/sig3/experiment/ontoedit"#North_America
```

Figure 12.19: Inferencing with Ontobroker in OntoEdit

Figure 12.20) obviously leads to less answers. Thus, only the fact that `New York` IS LOCATED IN `USA` is derived.



```
FORALL X,Y <- X:#City[#located_in->>Y].
```

```
oedit"#New_York;       "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
oedit"#Washington;     "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
oedit"#San_Francisco;  "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
oedit"#Seattle;        "http://www.ontoweb.org/sig3/experiment/ontoedit"#USA
oedit"#London;         "http://www.ontoweb.org/sig3/experiment/ontoedit"#Eng
oedit"#Paris;          "http://www.ontoweb.org/sig3/experiment/ontoedit"#Fra
oedit"#Madrid;         "http://www.ontoweb.org/sig3/experiment/ontoedit"#Spa
oedit"#Cairo;          "http://www.ontoweb.org/sig3/experiment/ontoedit"#Egy
```

Figure 12.20: Switching axioms off in OntoEdit

12.3.4 Main Lessons Learned in EON2002

We conclude this chapter by summarizing the main lessons learned in the EON2002 workshop.

- There exist at least three main categories in which tools can be divided: (i) Frame-oriented tools, (ii) Description Logic-oriented tools and (iii) Natural Language Processing-oriented tools. From the experiment descriptions and discussions at the workshop we derived the following main distinctions of tools and related modelling habits of their users. Quite naturally, the usage of a particular type of tool typically reflected the background of its user since most of the participants in the experiment were members of the corresponding tool developing crews.

- Frame-oriented tools included OntoEdit (Sure, 2002c), Protégé (Noy, 2002), SemTalk (Fillies, 2002) and WebODE (Corcho et al., 2002). Their users tried to stay as close as possible to the given domain description while modelling the domain ontology (*cf.* Section 3.2) according to the description. *E.g.* they used as concept names the concepts given in the description and modelled only very few additional concepts, that were in the text not explicitly stated.
- Description Logic-oriented tools included OilEd (Bechhofer, 2002), Loom (Gangemi, 2002) and OpenKnoME (Rogers, 2002). Most noteworthy was, that DL models require a rather complete covering of a domain. Therefore some users tended to model numerous additional concepts. One included numerous concepts from a top-level ontology (*cf.* Section 3.2). Another one modelled a detailed description of available transport vehicles and included atypic forms of vehicles such as amphibious vehicles, flying cars or water taxis.
- Natural Language Processing-oriented tools included TERMINAE (Aussenac-Gilles et al., 2002). An important aspect for their users is that the meaning of words and phrases is specific to a domain and can be inferred by observing the regularities of their use (in documents for instance). Thus, they also tend to model those concepts given in the description. The main interest is not to provide a powerful representation but rather the linkage of an ontology and corresponding texts. *E.g.* for given terminological forms the different occurrences of the corresponding terms in the text are given as well as an enumeration of used synonyms. For instance, the terminological form `meansOfTransport` gives the different occurrences of the term in the text, and it says that other terms in the text, "kind of transport" and "kinds of transportation", are used as synonyms (this example is taken from (Aussenac-Gilles et al., 2002)).

• Another important result of the discussion was regarding the use of the modelled ontologies as a backbone for application systems, *e.g.* a tourist booking system. From a systems point of view, the derived models imply different implementations based on the different notions for modelling constraints. The focus of the discussion was on the distinction between Frame-based and Description Logic-based ontologies, since they presented the majority in the experiment.

- In the Frame-based ontologies the constraints were modelled by using predicates that are "true" or "false" (*cf.* Section 12.3.3). A violation of these constraints would typically result in an error message, that can be displayed to users. *E.g.* when considering the constraint *"it is not possible to go from America to Europe by train, car, bike or motorbike"*, one would get an error message if telling the system that one travels by car from Europe to America. In Description Logic-based models, the model itself does not allow for such facts. However, during the discussion it was unclear how systems based on such ontologies would handle this.

- In Frame-based ontologies the concept hierarchy is explicitly defined whereas in Description Logic models the hierarchy is derived by an inference engine by using subsumption and might change whenever definitions for concepts change. Compared on a qualitative level, Frame-based ontologies are therefore likely to be more stable in their taxonomical structure than Description Logic-based ontologies. When using ontologies for navigational purposes, the Frame-based ontologies therefore seem to be more user-friendly, since users like to follow known paths through navigational structures.

In the end all participants found the discussion and the results very fruitful and agreed that this workshop as a starting point should lead to more formal evaluations of such tools.

12.3.5 Future Directions for EON Experiments

It is planned to continue with performing experiments in different directions. A roadmap for further experiments was set up[24] considering different dimensions that need to be evaluated. Thereby the more general evaluation framework for technology oriented evaluation presented in Section 4.6 was refined particularly for OEEs.

We will briefly describe the *objectives* of the experiments, *what* needs to be done to set up the experiments, *who* are the intended participants and what is the expected *output*.

Experiment 1. Evaluation of knowledge models

- *Objective*
  - To analyze which aspects can be and cannot be represented in each tool
  - To analyze how different ontology aspects must be represented in each tool
- *What*
  - Provide problem description in a domain that must be modelled in each tool

[24]The author would like to thank Oscar Corcho, Asunción Gómez-Pérez and Natalya F. Noy, who initiated this roadmap for future experiments. It was presented by Oscar Corcho at the EON 2002 workshop.

  - What characteristic does the description needs to fulfill?
    - Clear and concise?
    - Ambiguous enough? → it allows exploiting the main features of each tool
  - No commitments about how ontologies should be modelled

- *Who*
  - KR[25] experts from each group use their own ontology development tool to model the problem

- *Output*
  - Conclusions and guidelines
    - Which components can be represented with each tool
    - Which information has not been represented
    - Which workarounds are needed in each tool to represent components that are not directly supported in their model (for instance, reification of n-ary relations, need for creating complex logical axioms to represent some pieces of information, etc.)
    - How to represent axioms or constraints
    - How to represent components that allow obtaining new knowledge

Experiment 2. Usability and edition functionality

- *Objective*
  - To analyze the usability of ontology tools.
    - Clarity and consistency of the user interface
    - Learning time
    - Stability, help system, *etc.*

- *What*
  - A description of a problem in a domain
  - We can reuse the description of experiment 1[26]

- *Who*

[25]"KR" stands for "Knowledge Representation".
[26]That means the description of the EON 2002 experiment´, *cf.* Section 12.3.2.

    - KR experts (usability, stability, clarity of the tool's knowledge model, etc.)
    - KR novices (learning)
        * How much training is needed?

- *Output*
    - Conclusions and guidelines
        * Which functionalities, knowledge models, etc., are easier to learn and use
        * How much knowledge was represented, how long it took, *etc.*

Experiment 3. Scalability

- *Objective*
    - To analyze hardware and time resources needed to manage large ontologies
        * Thousands of components
- *What*
    - Select the set of ontologies
        * Mainly concept taxonomies (*e.g.*, UNSPSC)
        * With many relationships between concepts: densely interconnected graphs (*e.g.*, WordNet, Cyc)
        * With many thousands of instances (*e.g.*, UMLS)
    - Which measures?
        * Time needed to open/save ontologies
        * Time needed to show the concept hierarchy
        * Time needed to create/update/remove ontology components
        * Time needed to compute simple/complex queries Memory resources needed
- *Who*
    - No special requirements on people
- *Output*
    - Conclusions and metrics
        * How large are the ontologies that can be managed in each tool

Experiment 4. Navigability

- *Objective*
  - To analyze how ontology tools allow navigating large ontologies
- *What*
  - Select the set of ontologies
    - As in previous experiment
  - Which measures?
    - How easy is it to search for a component (graphically, text based, *etc.*)
    - How easy is it to extend the ontology with new components (concepts, instances, relations, *etc.*)
      - The user must first have an idea about where are the terms he needs, how to extend them, whether what he is trying to do has been already represented somewhere else or not, *etc.*
    - How easy is it to obtain a small part of the ontology
    - How easy is it to integrate components from different ontologies
    - How easy is it to read the documentation provided (in whatever formats it is provided by the tool)
    - *Etc.*
- *Who*
  - No special requirements on people
- *Output*
  - Conclusions and metrics
    - How large are the ontologies that can be navigated in each tool
    - How well did tools perform with respect to the above mentioned measures

Experiment 5. Interoperability

- *Objective*
  - To analyze how ontologies can be exchanged (exported and/or imported) between:
    - Tools

        · Ontology-related tools
        · General knowledge & software-engineering tools
    * Ontology languages

- *What*
  - Select the set of ontologies
    - The ontologies developed in experiment 1[27]
- *Who*
  - KR experts to evaluate results
- *Output*
  - Conclusions, metrics and guidelines:
    - Quality of exportations and importations
    - Interoperability
    - How exported/imported ontologies can be integrated in different systems
    - In which cases it is better to use one ontology tool or another for different domains and with different modelling/reasoning needs.

Experiment 6. Other functionalities

- *Objective*
  - To create a list of functionalities of each tool

The proposal for the next EON 2003 workshop is already submitted (*cf.* (Sure et al., 2003b)), it is currently being discussed which of the described experiments will be carried out as part of the workshop.

[27]That means the description of the EON 2002 experiment, *cf.* Section 12.3.2

Part V

Related Work & Conclusions

"The important thing is not to stop questioning."
— Albert Einstein

13 Related Work

Overview

This part contains (i) **related work** and (ii) **conclusions**.

In **this chapter** we present related work on methodologies in Section 13.1, on tools in Section 13.2 and on related case studies in Section 13.3.

13.1 Related Work on Methodologies

A first overview on methodologies for ontology engineering can be found in (Fernández-López, 1999). More recently, there have been joint efforts of OntoWeb members (*cf.* Chapter 12), who produced an extensive state-of-the-art overview of methodologies for ontology engineering (*cf.* (Gómez-Pérez et al., 2002b; Fernandéz-López et al., 2002)). There exist also deliverables on guidelines and best practices for industry (*cf.* (Léger et al., 2002a; 2002b)) with a focus on applications for E-Commerce, Information Retrieval, Portals and Web Communities. With respect to this work, especially the following approaches are noteworthy.

Since we believe that one of the main strong points is the tool support we offer for our methodology, we already give pointers to tools in the next section, whenever tool support is available for a methodology.

CommonKADS (Schreiber et al., 1999) is not *per se* a methodology for ontology development. It covers aspects from corporate knowledge management, through knowledge analysis and engineering, to the design and implementation of knowledge-intensive information systems. CommonKADS has a focus on the initial phases for developing knowledge management applications, we therefore relied on CommonKADS for the early feasibility stage. *E.g.* a number of worksheets is proposed that guide through the process of finding potential users and scenarios for successful implementation of knowledge management. CommonKADS is supported by PC PACK, a knowledge elicitation tool set, that provides support for the use of elicitation techniques such as interviewing, *i.e.* it supports the collaboration of knowledge engineers and domain experts.

Cyc (Lenat & Guha, 1990) arose from experience of the development of the Cyc knowledge base (KB)[1], which contains a huge amount of common sense knowledge. Cyc has been used during the experimentation in the High Performance Knowledge Bases (HPKB), a research program to advance the technology of how computers acquire, represent and manipulate knowledge[2]. Until now, this methodology is only used for building the Cyc KB. However, Cyc has different micro-theories showing the knowledge of different domains from different viewpoints. In some areas, several micro-theories can be used, and each micro-theory can be seen from different perspectives and with different assumptions. The Cyc project strongly enhanced the visibility of the knowledge engineering community, but at the same time it suffered from his very high goal to model "the world". Recently this goal has been lowered and now one has divided this too complex task into smaller ones,*e.g.* the Cyc top-level ontology was separated.

Recently, the **DOGMA** modelling approach (Jarrar & Meersman, 2002; Spyns et al., 2002a) has been presented. The database-inspired approach relies on the explicit decomposition of ontological resources into *ontology bases* in the form of simple binary facts called lexons and into so-called ontological commitments in the form of description rules and constraints. The modelling approach is implemented in the DOGMA Server, *e.g.* underlying (jointly with AIFB technology) the OntoWeb Semantic Portal (*cf.* Chapter 12), and accompanying tools such as the DOGMAModeler tool set.

The **Enterprise Ontology** (Uschold & King, 1995) (Uschold et al., 1998) proposed three main steps to engineer ontologies: (i) to identify the purpose, (ii) to capture the concepts and relationships between these concepts, and the terms used to refer to these concepts and relationships, and (iii) to codify the ontology. In fact, the principles behind this methodology influenced many work in the ontology community and they are also reflected in the steps kickoff and refinement of our On-To-Knowledge Methodology and extended them. Explicit tool support is given by the Ontolingua Server, but actually these principles heavily influenced the design of most of the more advanced ontology editors.

The **KACTUS** (Bernaras et al., 1996) approach requires an existing knowledge base for the ontology development. They propose to use means of abstraction, *i.e.* a bottom-up strategy, to extract on ontology out of the knowledge base as soon as an application in a similar domain is built. There is no specific tool support known for this methodology.

METHONTOLOGY (Gómez-Pérez, 1996; Fernández-López et al., 1999) is a methodology for building ontologies either from scratch, reusing other ontologies as they are, or by a process of re-engineering them. The framework enables the construction of ontologies at the "knowledge level". The framework consists of: identification

[1]Cyc knowledge base, see `http://www.cyc.com`
[2]HPKB, see `http://reliant.teknowledge.com/HPKB/about/about.html`

of the ontology development process where the main activities are identified (evaluation, configuration, management, conceptualization, integration implementation, *etc.*); a lifecycle based on evolving prototypes; and the methodology itself, which specifies the steps to be taken to perform each activity, the techniques used, the products to be output and how they are to be evaluated. METHONTOLOGY is partially supported by WebODE. Our combination of the On-To-Knowledge Methodology and OntoEdit is quite similar to the combinations of METHONTOLOGY and WebODE. In fact, they are the only duet that has reached a comparable level of integration of tool and methodology. However, we not only cover the development of ontologies, but have a much broader spectrum and include *e.g.* also the subsequent *usage* of ontology-based applications in our methodological framework. Also the support for collaboration is much stronger in the On-To-Knowledge Methodology and OntoEdit.

SENSUS (Swartout et al., 1997) is a top-down and middle-out approach for deriving domain specific ontologies from huge ontologies. The methodology is supported by Ontosaurus. The approach does not cover the engineering of ontologies as such, therefore offers a very specialized methodology.

TOVE (Uschold & Grueninger, 1996) proposes a formalized method for building ontologies based on competency questions. We found the approach of using competency questions, that describe the questions that an ontology should be able to answer, very helpful and integrated it in our On-To-Knowledge Methodology.

13.2 Related Work on Tools

An early overview of tools that support ontology engineering can be found in (Duineveld et al., 2000). However, like in the previous section on related work on methodologies, there have been recently joint efforts of OntoWeb members (*cf.* Chapter 12), who provided an extensive state-of-the-art overview on ontology related tools, including Ontology Engineering Environments (*cf.* (Gómez-Pérez et al., 2002a)). A sign for the growing interest in Ontologies and tools that support ontology engineering is the (also recently) published comparison of ontology editors on XML.com (*cf.* (Denny, 2002)). An evaluation of ontology engineering environments has been performed as part of the EON 2002 workshop (*cf.* Section 12.3 and (Sure & Angele, 2002)). With respect to this work, especially the following tools are noteworthy.

APECKS (Tennison & Shadbolt, 1998) is targeted mainly for use by domain experts, possibly in the absence of a knowledge engineer, and its aim is to foster and support debate about domain ontologies. It does not enforce consistency nor correctness, and instead allows different conceptualisations of a domain to coexist.

Chimaera (McGuinness et al., 2000a) is primarily a merging tool for ontologies. It contains only a simple editing environment and relies on the Ontolingua Server for more advanced modelling.

The **DOGMAModeler** is a set of tools for ontology engineering. It relies on ORM (*cf.*, *e.g.*, (Halpin, 2001)) as graphical notion and its cross-bonding ORM-ML to ensure easy exchange (*cf.* (Demey et al., 2002)). It supports the database-inspired DOGMA ontology engineering approach and is coupled with the DOGMA Server as a backend.

KAON OImodeller (Motik et al., 2002; Bozsak et al., 2002) belongs to the KAON tool suite that is currently evolving in collaboration with our partner institute FZI. The system is designed to be highly scalable and relies on an advanced conceptual modelling approach that balances some typical trade-offs to enable a more easily integration into existing enterprise information infrastructure. Compared to this work, the KAON tool suite does not provide such a range of specialized support for collaboration and inferencing. However, since both approaches rely on the OntoMat plugin framework (but they differ in their internal data structures) it is foreseen to integrate the two approaches to make *e.g.* the methodology plugins exchangeable.

OilEd (Bechhofer et al., 2001)is a graphical ontology editor that is dedicated to modelling of DAML+OIL ontologies. Thus, on the one hand it is dependent on a particular representation language, but on the other hand offers strong support for modelling such ontologies. A key aspect of OilEd is the use for FaCT (Horrocks, 1998) to classify ontologies and check consistency via translation from DAML+OIL to the SHIQ description logic. However, the tool is not extensible *e.g.* by plugins, nor does is provide sophisticated support for collaboration aspects.

The **Ontolingua** (Farquhar et al., 1996) Server is a set of tools and services that support the building of shared ontologies between distributed groups. It provides access to a library of ontologies and translators to languages such as Prolog, CLIPS and Loom. The set of tools was one of the first sophisticated ontology engineering environments with a special focus on the collaboration aspects. However, the development has not kept pace with the evolving current standards such as RDF or DAML+OIL, nor with the state-of-the-art technology.

Ontosaurus (Swartout et al., 1996) consists of two modules: an ontology server, which uses Loom as knowledge representation system, and an ontology 'browser server' that dynamically creates HTML pages to display the ontology hierarchy. Translators exist from Loom to Ontolingua, KIF, KRSS and C++. Similar to the Ontolingua Server, it was a milestone in the development of OEEs, but the development has not kept pace with the evolving standards and technologies.

Protégé (Noy et al., 2000) is a well established ontology editor with a large user community. The design of the tool is very similar to OntoEdit since it actually was the first editor with an extensible plugin structure and it also relies on the frame paradigm for modelling. Numerous plugins from external developers exist. It also supports current standards like RDF(S) and DAML+OIL. Recently also support for axioms was added through the "PAL tab" (Protégé axiom language, *cf.* (Hou et al., 2002)). However, compared to OntoEdit, the inferencing support is not as strong since Ontobroker is fully integrated into OntoEdit. OntoEdit also has advanced capabilities for the distributed development of ontologies. In a nutshell, both tools are quite equal, but each tool has its strengths on particular areas.

WebODE (Arpírez et al., 2001) is an "ontology engineering workbench" that provides various service for ontology engineering. Similar to OntoEdit, it is accompanied by a sophisticated methodology of ontology engineering, *viz.* METHONTOLOGY (*cf.* (Gómez-Pérez, 1996; Fernández-López et al., 1999)). In contrast to OntoEdit and Protégé it (both Java standalone applications) is purely web-based and is built on top of an application server. At the same time this gives WebODE an equal level of extensibility. For inferencing services it relies on Prolog. It provides translators to current standards such as RDF(S) and DAML+OIL. OntoEdit has particularly more advanced support for collaboration through specialized plugins and a better integration of the inferencing capabilities. OntoEdit also makes use of inferencing for various purposes. In a nutshell, WebODE is comparable to OntoEdit and Protégé, but again, each tool has its strength on particular areas.

WebOnto (Domingue, 1998) and the accompanying tool **Tadzebao** support graphical ontology engineering and in particular the argument between users on the ontology design, using text, GIF images and even hand drawn sketches. The strength of this approach lies in the advanced support for communication between ontology engineers and domain experts. However, the tool is not extensible nor does it provide sophisticated and specialized inferencing support.

13.3 Related Work on Case Studies

The growing interest in the Semantic Web is explicated in a growing number of projects in that area. Again, similar to the sections on related work on methodologies and tools, there exist joint efforts of OntoWeb members (*cf.* Chapter 12), who provided an extensive state-of-the-art overview on projects, business scenarios, initiatives and case studies (*cf.* (Gómez-Pérez et al., 2002b; Léger et al., 2002a; Brown et al., 2002)). With respect to this work, especially the following initiatives are noteworthy.

The **DARPA–DAML** programme[3] is a huge effort in the US that actually combines numerous projects in the area of the Semantic Web. The project has similar objectives as the On-To-Knowledge project (*cf.* Chapter 11). As sketched in Section 6.1, their efforts for designing representation languages, *viz.* DAML and OIL, were joint into

[3]Darpa Agent Markup Language, see `http://www.daml.org/`

DAML+OIL. In general, there exist strong links and numerous collaborations between DAML and On-To-Knowledge/OntoWeb.

KA2 (Benjamins & Fensel, 1998; Benjamins et al., 1999) was an initial effort that formed the community around "Knowledge Acquisition". As such it is the predecessor of OntoWeb and many partners from KA2 are now continuing their collaboration within OntoWeb. As shown in Section 12.2, the community portal of KA2 provided ideas and technology that is re-used in the OntoWeb Semantic Portal.

With respect to our conceptual architecture for SEmantic portALs, SEAL, we would like to mention the following related work. Given the aforementioned difficulties with managing complex Web content (*cf.* Section 12.2.1), several approaches tried to facilitate database technology to simplify the creation and maintenance of data-intensive web-sites. Systems, such as ARANEUS (*cf.* (Mecca et al., 1999)) and AutoWeb (*cf.* (Ceri et al., 2000)) also take a declarative approach. In contrast to SEAL that relies on standard Semantic Web technologies these systems introduce their own data models and query languages, although all approaches share the idea to provide high-level descriptions of web-sites by distinct orthogonal dimensions. The idea of leveraging mediation technologies for the acquisition of data is also found in approaches like Strudel (*cf.* (Fernandez et al., 2000)) and Tiramisu (*cf.* (Anderson et al., 1999)), they propose a separation according to the aforementioned task profiles as well. Strudel does not concern the aspects of site maintenance and personalization. It is actually only an implementation tool, not a management system. Closest to our approach come OntoWebber (*cf.* (Y. Jin, 2001)) and IIPS, the "Intelligent Information Presentation System", (*cf.* (Lei et al., 2002)), but both concentrate more on the aspect of modelling explicit site models that include also layout information. From our point of view the SEAL framework and it's application as the OntoWeb portal is rather unique with respect to the collection of methods used and the functionality provided.

14 Conclusions

In **this chapter** we summarize the main contributions of this work in Section 14.1 and, last but not least, present an outlook to future work and research directions in Section 14.2.

14.1 Summary

The **On-To-Knowledge Methodology** and its **accompanying tool support** provide an advanced framework for engineering ontology based applications that is highly re-usable for different kinds of domains. We integrated, adapted and extended existing (similar) methodologies. The On-To-Knowledge Methodology is based on the separation of two processes:

(i) the **Knowledge Meta Process** for developing and employing ontology based knowledge management applications, and

(i) the **Knowledge Process** for running them.

The accompanying tools are an extension of the already existing ontology engineering environment OntoEdit and provide specialized support for steps of the Knowledge Meta Process.

We presented the application of the On-To-Knowledge Methodology and its accompanying tool support in a **large spectrum of practical scenarios** for ontology based knowledge management applications. The scenarios and implementations range

(i) from **corporate intranets** to **the Web**,

(ii) from **industry** to **academia**, and

(iii) from **prototypes** to **productive systems**.

The presented work has been performed in the EU project On-To-Knowledge and the EU thematic network OntoWeb, both having a large visibility in the Semantic Web community. The lessons learned in On-To-Knowledge are (in a nutshell):

- **Early ontology development** is often unstructured **brainstorming** rather than careful design.
- **Different processes drive KM projects**, but "Human Issues" might dominate other ones.
- **Guidelines for domain experts in industrial contexts** have to be **pragmatic**, otherwise they are unlikely to be understood and to be used at all.
- **Collaborative ontology engineering** requires **physical presence *and* advanced tool support**.

The lessons learned in OntoWeb are (in a nutshell):

- There currently exist (at least) three main categories in which ontology engineering tools can be divided:
  1. **Frame**-oriented tools: their users tend to model ontologies as close as possible to a given natural language domain description.
  2. **Description Logic**-oriented (DL) tools: in comparison to the Frame-oriented tools, these users tend to model numerous additional concepts to get a rather complete coverage of a domain.
  3. **Natural Language Processing**-oriented tools: their users have a main interest in providing a linkage of an ontology and corresponding texts. Thus, in comparison to Frame- and DL-oriented tools, their aim is not to provide a powerful representation.
- Applications that are built on top of ontologies and that are modelled with such tools have the following specialities:
  1. In the Frame-based ontologies constraints are modelled by using predicates that are "true" or "false". A violation of these constraints would typically result in an error message, that can be displayed to users. In Description Logic-based models, the model itself does not allow for such facts.
  2. In Frame-based ontologies the concept hierarchy is **explicitly defined** whereas in Description Logic models the hierarchy is **derived by an inference engine** by using subsumption and might change whenever definitions

for concepts change. Compared on a qualitative level, Frame-based ontologies are therefore likely to be **more stable** in their taxonomical structure than Description Logic-based ontologies. When using ontologies for navigational purposes, the Frame-based ontologies therefore seem to be **more user-friendly**, since users like to follow known paths through navigational structures.

The following outlook presents future extensions and indicates already existing impact of this work on other projects.

14.2 Impacts & Outlook

We finally conclude by presenting some impact that this work already has on current research projects combined with a future outlook on the upcoming research agenda for ontology based knowledge management.

Tighter Integration of Mind Maps

The usage of mind maps™ for the initial stages of ontology engineering (by using the Mind2Onto framework, *cf.* Sections 7.2, 11.4 and 11.5) has already successfully been reused in other projects, *e.g.* in the German thematic network "Kompetenznetzwerk Wissensmanagement"[1]. What is needed next is the tighter technical integration which enables a round-tripping between mind maps™ and ontologies to overcome the modelling gap in between. Ideally, domain experts could create and maintain mind maps™ while ontology engineers create and maintain accompanying ontologies. Elements of such mind maps™ would need to be dynamically mapped into an ontology, changes in one of the entities would result *e.g.* in a structured To-Do list for the other entity.

User-friendly Implementation of OntoClean

Though a mature methodology like OntoClean (*cf.* (Guarino & Welty, 2002)) exists for the evaluation of ontologies, the research field is still rather immature in terms of applicability. The implementation of OntoClean in OntoEdit (*cf.* Section 9.3) was a first step to operationalize the methodology. Still, a typical ontology engineer without special training will unlikely be able to apply the methodology efficiently and effectively. Therefore, a more user-friendly implementation of OntoClean is needed, that encapsules the philosophical notions into understandable tasks. First discussions with

[1]Kompetenznetzwerk Wissensmanagement, see `http://wiman.server.de`

some of the authors of the OntoClean methodology based on our initial implementation resulted in the idea to create a catalog of questions that helps classifying concepts according to the taxonomy of properties. However, a clear design concept is on the upcoming research agenda.

Integrating Manual Ontology Engineering and Ontology Learning

In this work we deal with the *manual* engineering of ontologies in contrast to *(semi-) automatic learning* of ontologies. *Ontology learning* (*cf.* (Maedche, 2002a)) is motivated by the need for quickly generated ontologies that reflect a large volume of content. The main advantage of applying machine learning techniques for ontology learning is seen in avoiding the "knowledge acquisition bottleneck" for the creation of large amounts of ontologies, some drawbacks are (i) the generation of rather weakly structured ontologies and (ii) missing methodological guidelines for integrating ontology learning in ontology engineering. A combination of manual engineering and automatic learning could (i) reduce the time needed during manual engineering and (ii) increase the level of quality gained during learning ontologies. A possible scenario for the tight integration of both approaches could be the manual creation of an initial ontology that is refined in alternating learning and manual sessions.

Combining Ontologies with Data Mining

The On-To-Knowledge Methodology has already been applied in other projects, *e.g.* the "SemiPort" project[2] that aims at combining ontologies with data mining techniques to close the loops of engineering ontologies and using them (*cf.* (Gonzalez-Olalla & Stumme, 2002)). 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 (*cf.* Section 11.4) it was initially planned to keep track of the usage patterns to provide suggestions which parts of an ontology might need modifications. However, the SemiPort project goes a step further and explores how to use such usage patterns for a more efficient searching in knowledge portals (*cf.* (Stojanovic et al., 2002d)) such as constituted by our SEAL approach (*cf.* Section 12.2).

Maturity Levels for Knowledge Management

The VISION project[3] will provide a strategic roadmap towards next-generation organisational knowledge management. The On-To-Knowledge Methodology and related

[2]http://km.aifb.uni-karlsruhe.de/semiport/
[3]EU IST-2002-38513 VISION project, see `http://www.fzi.de/vision/`

case studies already served as a valuable input for the collection of best KM practices in the "VISION" project. However, in this work we only partially covered aspects of introducing knowledge management into organizations (*cf.* Section 4.1). As part of the work in VISION, a holistic approach for the introduction and improvement of KM in organizations is developed (*cf.* (Hefke & Trunko, 2002)) with a special focus on guidelines for enabling ambient access to knowledge within next-generation applications.

Making the Semantic Web a Success

A key issue for making the Semantic Web a large success story is to make the creation of meta data as easy as possible. On the one hand that requires support for the (semi-) automatic annotation of textual documents and the "deep web", *i.e.* databases, on the other hand Semantic Web technologies need to be integrated into sophisticated applications (*cf.*, *e.g.*, (Smolle & Sure, 2002)) and, even more important, into standard products (*cf.*, *e.g.*, (Fillies & Sure, 2002)).

Ideally, such technologies are deeply embedded into the daily working desktop (*cf.* (Staab & Schnurr, 2002)).

Finally, knowledge workers will not perceive technologies anymore, but are used to **receive the right answers at the right time in the right context**.

Part VI

Appendix

'Synonyms, ordered by estimated frequency' search for noun 'appendix':

Sense 1
appendix – *(supplementary material that is collected and appended*
at the back of a book)
=> *addendum, supplement, postscript – (textual matter that*
is added onto a publication; usually at the end)

Sense 2
appendix, vermiform appendix, vermiform process, cecal appendage – *(a vestigial process that extends from the lower end*
of the cecum and that resembles a small pouch)
=> *process, outgrowth, appendage – (a natural prolongation*
or projection from a part of an organism either
animal or plant; 'a bony process')

— Derived from WordNet 1.7.1, Copyright 2001 by Princeton University.
All rights reserved.

A OntoEdit Outside

Overview

This appendix provides (i) a detailed view on OntoEdit from a users perspective (it therefore extends the description OntoEdit in Section 6.2) and (ii) the example ontology that was used as a running example in Parts II and III. In this chapter, we start the more detailed description of OntoEdit by depicting the historical evolution of OntoEdit and then illustrate step by step the basic functionalities of OntoEdit from a users perspective. The next Chapter B focusses on the internal aspects of OntoEdit, *viz.* the implementation of the OntoMat framework. Then, in Chapter C we present the DAML+OIL features, that OntoEdit supports. Finally, we present the example ontology in Chapter D.

History & Facts

The development of OntoEdit was initiated at the Institute AIFB by Alexander Maedche (now: Research Center FZI, Research Group WIM, Karlsruhe) and Dirk Wenke (now: chief developer of OntoEdit at Ontoprise GmbH, Karlsruhe, Germany), and is currently being continued by the constant efforts of Dirk. OntoEdit evolved during the last years from the status of university prototype to commercial product and is now distributed by the company Ontoprise GmbH[1].

During the course of the On-To-Knowledge project (see Chapter 11) OntoEdit was re-implemented by instantiating the OntoMat plugin framework. Several additional plugins for OntoEdit were developed driven by the requirements coming from the case studies in On-To-Knowledge to support different steps of the methodology (see Chapters 4 and 5). These plugins will be described in the following sections. The experiences made with OntoEdit within the project resulted in valuable feedback and guided further directions of OntoEdit's development (*e.g.* bug fixes, requirements, functionalities).

[1]http://www.ontoprise.de

OntoEdit is developed purely in Java. There exist several distributions aiming at different audiences. *E.g.* while the freely available *OntoEdit Free Edition* offers basic functionalities, the *OntoEdit Inferencing Edition* provides also sophisticated inferencing and, *e.g.* visualization. The following descriptions are based on OntoEdit version *2.6*.

We will now briefly introduce the basic functionalities of OntoEdit.

Main GUI

Figure A.1 depicts the main elements of the GUI. The different menu items on top will be described in the following sections. The buttons represent short cuts to functionalities provided in the menus like "New ontology" or "Save ontology".

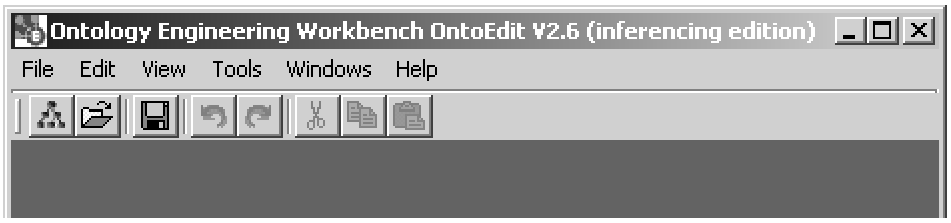

Ontology Engineering Workbench OntoEdit V2.6 (inferencing edition)
File
Edit
View
Tools
Windows
Help

Figure A.1: The main GUI of OntoEdit

File Menu

The "File" menu (*cf.* Figure A.2) allows for the following actions:

- **New ontology:** *cf.* Section A
- **Open ontology:** Allows for browsing through local directories to open existing ontologies.
- **Import / Export:** One may import and export ontologies in numerous representation languages like F-Logic (Kifer et al., 1995), RDF(S) (Brickley & Guha, 2002), DAML+OIL (Horrocks et al., 2001) and in future OWL (Smith et al., 2002). Future evolving formats may be supported by plugins for OntoEdit (*cf.* Section 6.3).
- **Save ontology / Save ontology as ...:** Usually ontologies are stored in OntoEdits internal format, *viz.* OXML (Ontoprise, 2002c). Other formats are supported by 'Import / Export'.

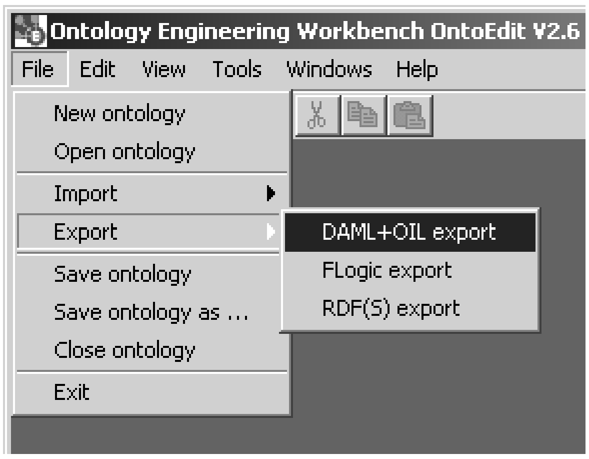
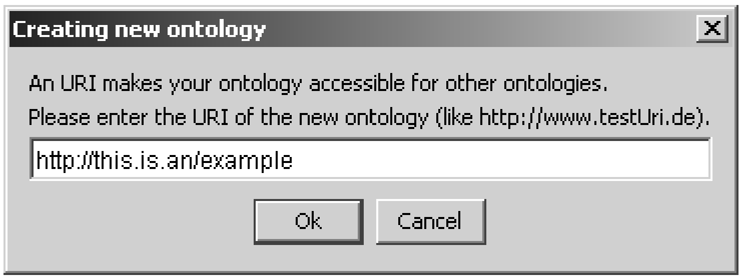


Figure A.2: File Menu

- **Close ontology:** Closes an open ontology window (an example for an ontology window is shown in the next section).
- **Exit:** Terminates the execution of OntoEdit.

New Ontology

To create a "New ontology" one first has to enter a URI that identifies this ontology uniquely (*cf.* Figure A.3), *e.g.* like `http://this.is.an/example`. If this ontology is to be put on the WWW, one might prefer to chose an URL where the ontology will be made available.


Creating new ontology

An URI makes your ontology accessible for other ontologies.
Please enter the URI of the new ontology (like http://www.testUri.de).

http://this.is.an/example

Figure A.3: Insert a URI

A new ontology window (*cf.* Figure A.4) opens showing several tabs. The "Concepts & Relations" tab is opened with just a root concept called DEFAULT_ROOT_CONCEPT.

We will now describe the tabs shown in Figure A.4 except for the tabs “General Axioms” and “Inferencing” that are described in the Sections 8.3 and 9.4.

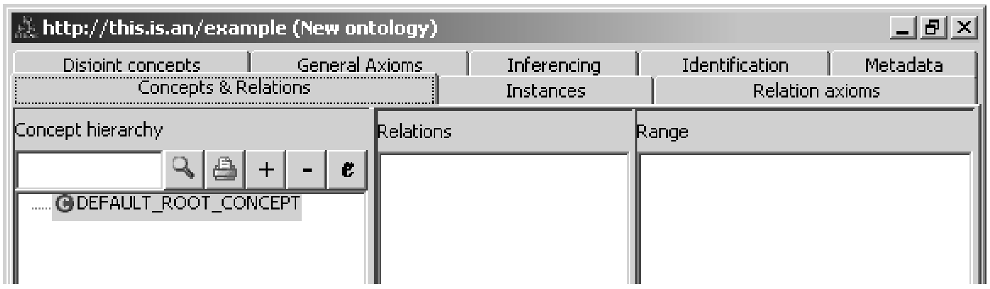


Figure A.4: Create a new ontology

Concepts & Relations

Possible actions in the “Concepts & Relations” tab are:

- Search for a specific concept (*cf.* Figure A.5)
  - To search for a specific concept you can enter the concept name to search for, *e.g.* “Person”. Finally click the lens button to search.

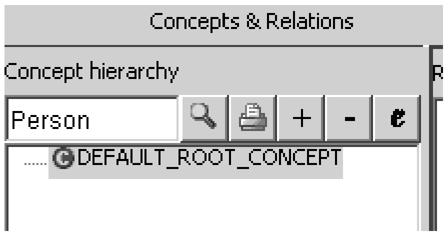


Figure A.5: Search for a specific concept

- Concept context menu (*cf.* Figure A.6)
  - To open the context menu one may right click a concept.
- Insert a new concept (*cf.* Figure A.7)
  - To insert a concept one may either click on the insert-button or select it from the popup-menu that appears after a right-mouse-button click. Then one may “windows-like” enter the identifier of the new concept.

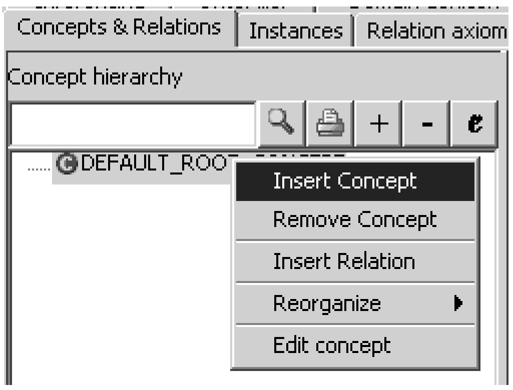


Figure A.6: Concept context menu

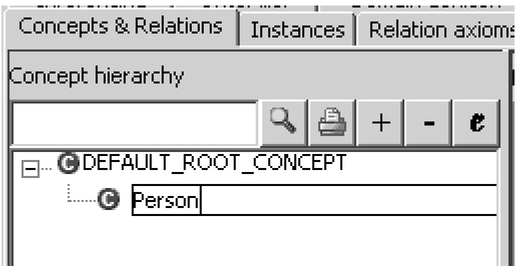


Figure A.7: Insert a new concept

- Delete a concept (not possible for the root concept)
  - Select a concept and press the minus button to delete it. One may also use the context menu (right click) to delete a concept.
  - All subconcepts of the selected concept will be deleted, too, if they don't exist in another tree-branch.
- Edit a concept / concept properties (*cf.* Figure A.8)
  - **External representation(s):** any name or number that identifies uniquely the new concept in a certain language.
  - **Instantiation:**
    - **concrete:** it may exist an object of the given concept.
    - **abstract:** the concept describes only an abstract concept and needs to be specified with a subconcept in order to get a concrete concept (*e.g.* the root concept).

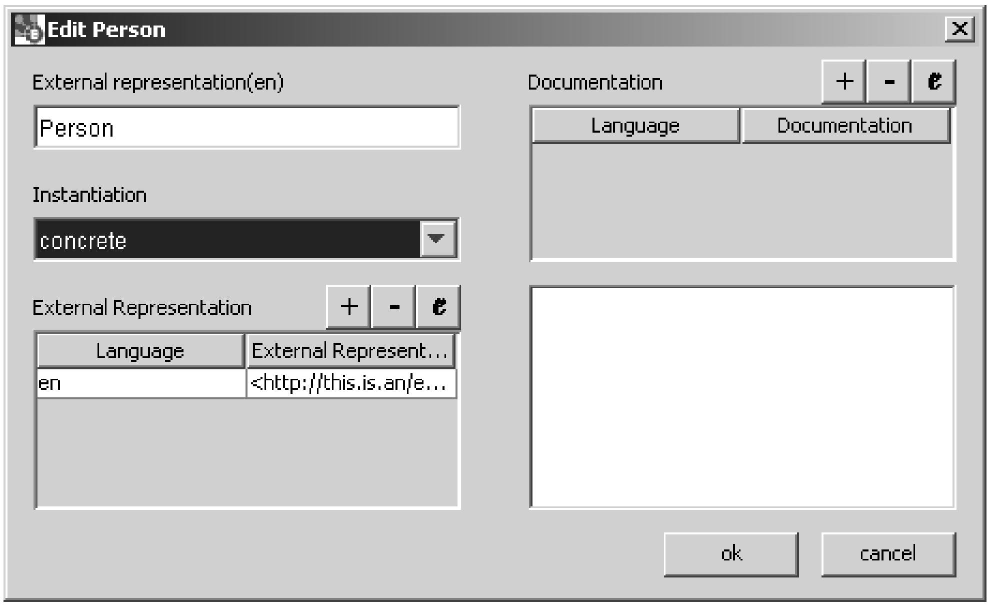


Figure A.8: Edit selected concept / concept properties

  - **Documentation:** you should document any assumptions and thoughts about the concept so other persons can understand and use the ontology, too. This can be done in multiple languages.
  - **Insert, Delete and Edit:** one may use the "+", "-" and "e" buttons to insert, delete and edit external representations and documentations.

- Reorganize – Copy/Move (similar to well-known "copy-and-paste" functionality)

  - This options allows for copying a concept as an new subconcept of another concept (multiple inheritance) or to move it to another placement in the concept-hierarchy. If one chooses one of these options, a dialog appears where one has to select the new superconcept.

- Relation context menu (*cf.* Figure A.9)

  - To open the context menu right click in the relation area or on a relation

- Insert a new relation (*cf.* Figure A.10)

  - To insert a new relation you need to select a concept first. Right click the concept to open it's context menu. Select 'insert relation' from the context menu.

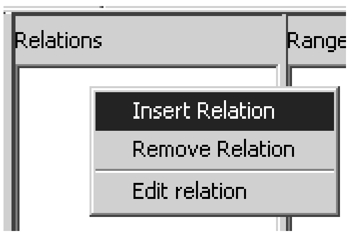
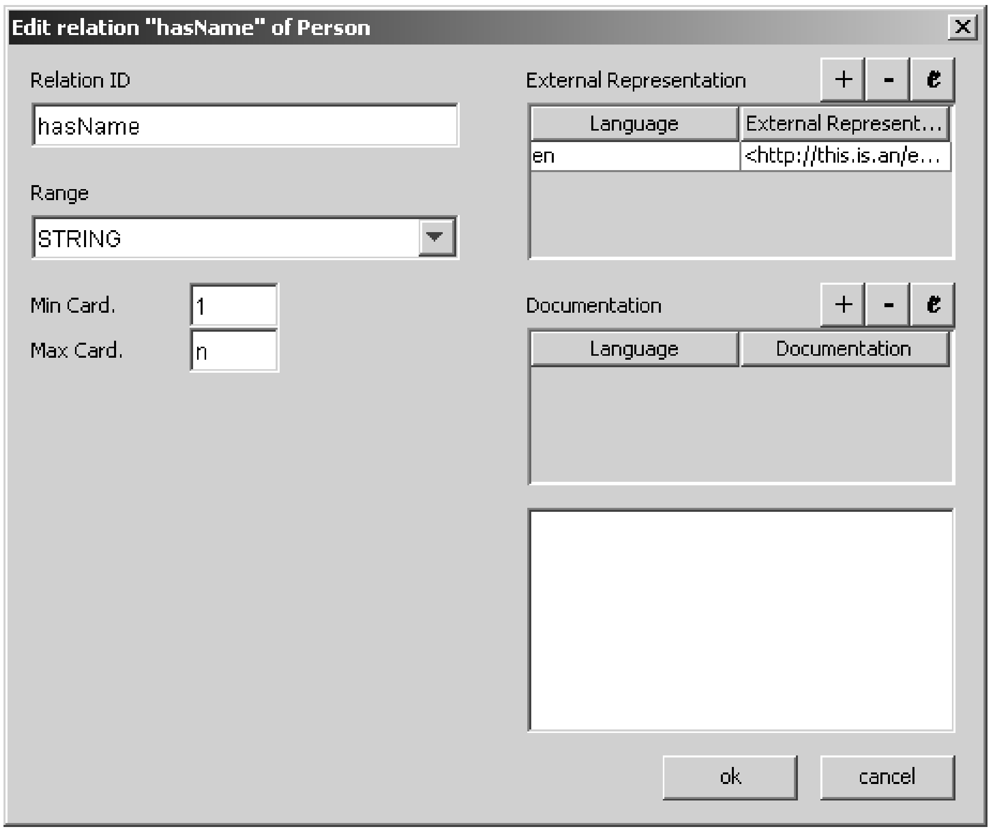


Figure A.9: Relation context menu



Relation ID: hasName

Range: STRING

Min Card. 1

Max Card. n

External Representation

Language	External Represent...
en	<http://this.is.an/e...

Documentation

Language	Documentation

Figure A.10: Insert new relation

* **Relation ID:** any name or number that identifies uniquely the new relation.
* **Range:** range of the relation (typically another concept, but the XML Schema data types (Biron & Malhotra, 2001) `String`, `Integer` and `Boolean` are supported for defining attributes)

* Some examples for defining ranges:
  - A concept Person has the relation COOPERATES WITH(PERSON, PERSON), so COOPERATES WITH has the range 'person' and tells with what other person (range) the person cooperates.
  - The relation OWNS CAR(PERSON, BOOLEAN) has the range BOOLEAN and says whether a person owns a car or not.
  - For relations inherited from superconcepts OntoEdit lets one refine / restrict the range of a relation. *I.e.* the superconcept Organization has a relation ORGANIZES with the range Events, the subconcept Institute inherits the relation ORGANIZES, but should be restricted to organizing Workshops. So, one may double click the relation (has a gray background, because it is inherited) in the relation view and change the range from event to workshop.
* **Min Card:**The minimum cardinality is described best with an example:
  - A Car usually has exactly one Car owner: 'Min Card' and 'Max Card' are set to '1'.
  - A Car owner has at least one Car (otherwise it wouldn't be a car owner): 'Min Card' is set to '1'.
  - A Person *e.g.* can OWN a Car, but doesn't need to own one: set 'Min Card' to '0'.
  - For relations inherited form superconcepts OntoEdit allows to refine / restrict the Min Card (see Range).
* **Max Card.:**
  - A Car owner can own more than one car so 'Max Card' is set to the maximum allowed number of cars, *i.e.* 'Max Card' set to '10'.
  - For relations inherited form superconcepts OntoEdit allows to refine / restrict the the Min Card. (see Range).
* **External Representation:** one may enter a relation name for external representation in different languages including special characters.
* **Documentation:** one should document any assumptions and thoughts about the relation so other persons can understand and use the ontology, too.
* **Insert, Delete and Edit:** this represents the usual functionality for external representations and documentations.

Instances Tab

Possible actions in the "Instances" tab are:

- Insert a new instance
- Delete an instance
- Edit an instance (instance properties)

Instances will be displayed for each concept. So if you select a concept in the concept hierarchy, its instances will be displayed in the right area. One should have in mind that all instances of a subconcept are also instances of the concept itself. That means that an instance of Student will be an instance of Person, too, if Student is defined as a subconcept of Person.

To see all instances of the ontology, one simply clicks on the root concept. If relations of an instance contain values, one may expand the instance node to see its values.

Instance Context Menu

To open the instance context menu right click in the instances area (*cf.* Figure A.11).

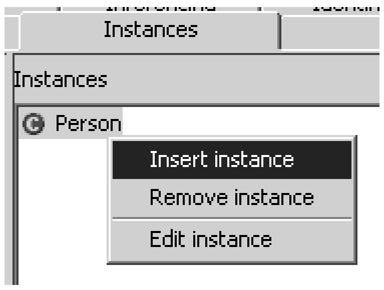
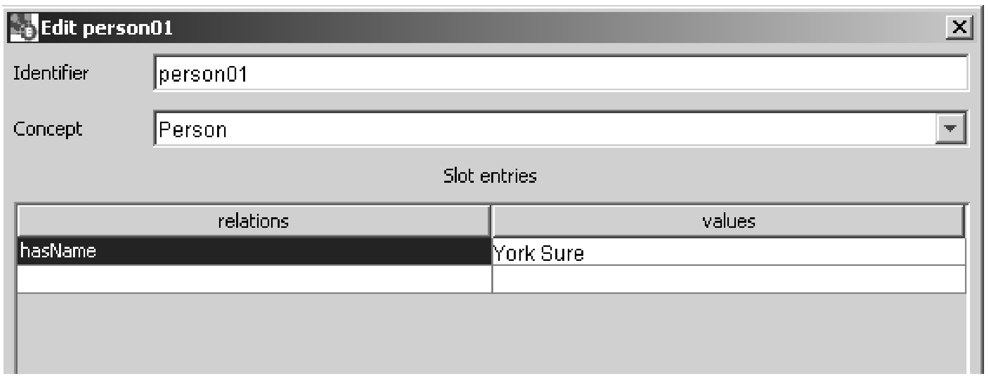


Figure A.11: Context menu for instances

- **Insert a new instance**: to insert a new instance one has to select it from the popup menu that appears after a right-mouse-button click. Then one may "windows-like" enter the identifier of the new instance.
- **Remove an instance**: to remove an instance one has to select the instance and then to select "remove instance" from the popup menu that appears after a right-mouse-button click.
- **Editing an instance**: selecting an instance and selecting "edit instance" from the popup menu enables the editing of it. The following dialog appears (Figure A.12), where one can select in the first column a relation and in the second column the value this instance should be related to. If the range of the relation

is `String`, `Integer` or `Boolean` one may enter the value in a text field, otherwise a combo box appears where one can select one of the possible instances.



Identifier: person01

Concept: Person

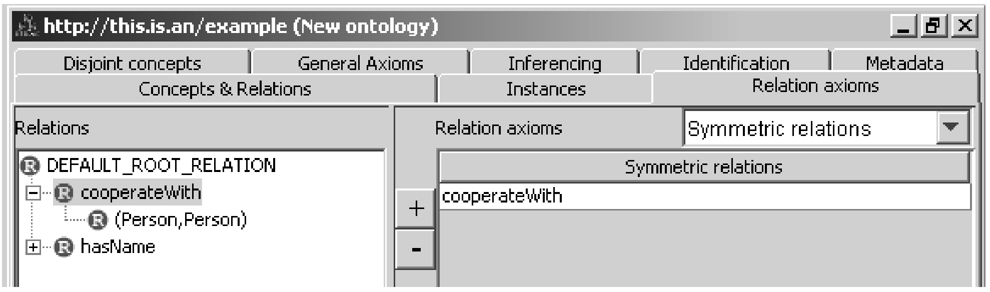
Slot entries

relations	values
hasName	York Sure

Figure A.12: Editing an instance

Relation Axioms Tab

The “Relation Axioms” tab is used to browse existing relations and to enter axioms for relations, that means special properties of relations and relations to other relations (*cf.* Figure A.13).



Relation axioms: Symmetric relations

Symmetric relations
cooperateWith

Figure A.13: Relation axioms tab

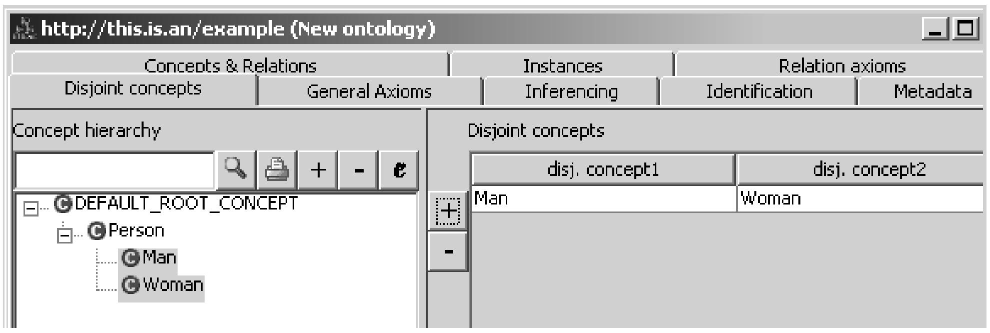
- Symmetric relation `a` R `b` $\Rrightarrow$ `b` R `a`
  - *I.e.* COOPERATES WITH(PERSON A, PERSON B)
    $\Rrightarrow$ COOPERATES WITH(PERSON B, PERSON A)
    If `Person A` cooperates with `Person B` it is true that `Person B` does also cooperate with `Person A`, therefore you might declare this relation to be symmetric.

- Transitive relation a R b and b R c $\Rrightarrow$ a R c
  - *I.e.* can communicate in English with(Person A, Person B) and can communicate in English with(Person B, Person C) $\Rrightarrow$ can communicate in English with(Person A, Person C) If `Person A` is able to communicate with `Person B` in English and `Person B` speaks English with Person C it is probably true that Person A is able to communicate in English with `Person C`.
- Inverse relation
  - *I.e.* slower than(Car, Car) is the inverse relation to faster than(Car, Car)
  - Relations can be defined as globally or locally inverse.

Disjoint Concepts Tab

The "Disjoint Concepts" tab allows to define disjoint concepts (*cf.* Figure A.14). One needs to multi-select concepts in the concept hierarchy an then click on the '+' button to define them as pairwise disjoint. The editor automatically generates the necessary pairs if one selects more than two concepts.

To delete such a pair one needs to select the row in the right window and press '–'. Or one may select it from the popup menu that appears after a right mouse click.

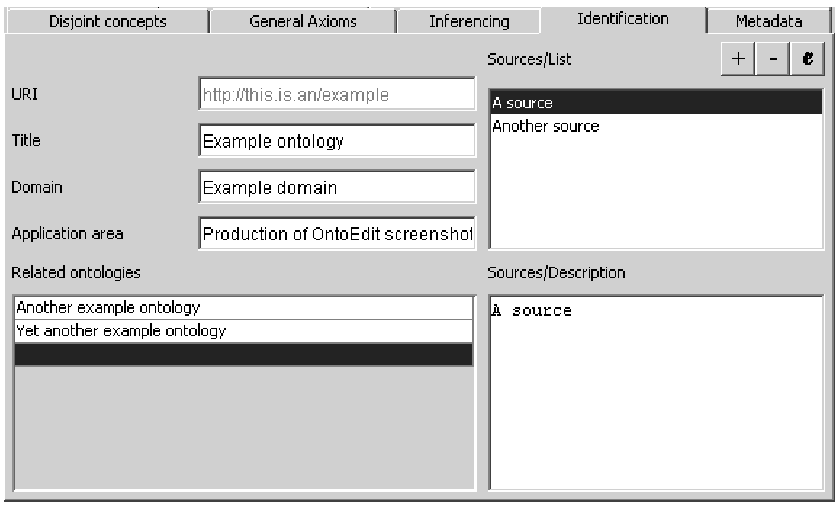


disj. concept1	disj. concept2
Man	Woman

Figure A.14: Disjoint concepts tab

Identification Tab and Metadata Tab

The two tabs "Identification" and "Metadata" (*cf.* Figures A.15 and A.16) offer the possibility of giving information about the ontology, *e.g.* the URI, authors, documentation, etc.



URI	http://this.is.an/example
Title	Example ontology
Domain	Example domain
Application area	Production of OntoEdit screenshot

Related ontologies

- Another example ontology
- Yet another example ontology

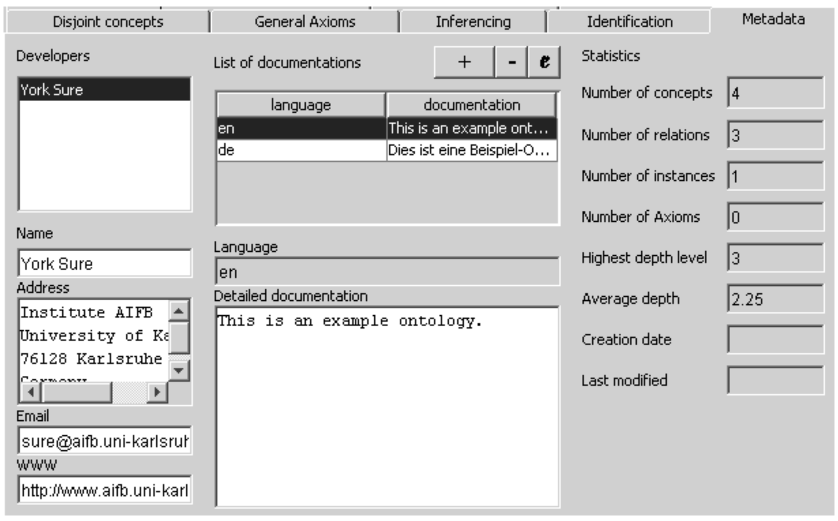
Sources/List

- A source
- Another source

Sources/Description

```
A source
```

Figure A.15: Identification tab



Developers

- York Sure

Name	York Sure
Address	Institute AIFB University of Ka 76128 Karlsruhe Germany
Email	sure@aifb.uni-karlsruh
WWW	http://www.aifb.uni-karl

List of documentations

language	documentation
en	This is an example ont...
de	Dies ist eine Beispiel-O...

Language	en
Detailed documentation	This is an example ontology.

Statistics

Number of concepts	4
Number of relations	3
Number of instances	1
Number of Axioms	0
Highest depth level	3
Average depth	2.25
Creation date	
Last modified	

Figure A.16: Metatada tab

Edit Menu

The “Edit” menu offers typical undo/redo, cut and copy-and-paste functionalities which are not yet fully provided. A similar functionality to copy-and-paste is provided by the reorganization of the concept hierarchy (*cf.* Section A).

View Menu

The "View" menu shown in Figure A.17 allows for the following actions:

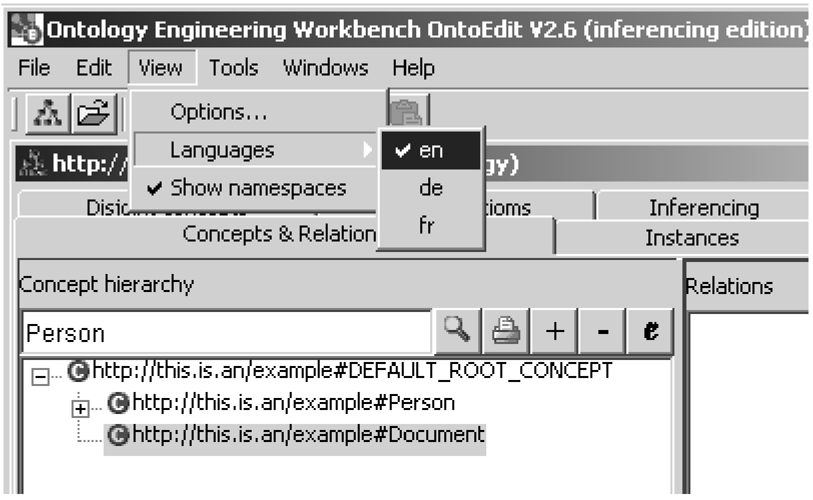

Ontology Engineering Workbench OntoEdit V2.6 (inferencing edition)
File
Edit
View
Tools
Windows
Help
Options...
Languages
✓ Show namespaces
✓ en
de
fr
Concepts & Relation
Inferencing
Instances
Relations
Concept hierarchy
Person
http://this.is.an/example#DEFAULT_ROOT_CONCEPT
http://this.is.an/example#Person
http://this.is.an/example#Document

Figure A.17: View menu

- **Options:** This represents the so-called "Options Manager" of the OntoMat framework (*cf.* Section 6.3). All preferences for plugins can be set and changed here, *e.g.* one might add languages that should be supported (*cf.* Figure A.18).
- **Languages:** OntoEdit supports external representations and documentations in multiple languages. By default English, German and French are supported. Switching to a language means to change the currently shown external representation of concepts and relations in the "Concept & Relations" tab (*cf.* Section A).
- **Show namespaces:** OntoEdit supports XML namespaces (Bray et al., 1999) for the naming of concepts, relations and instances. This options switches on/off the visualization of namespaces in the GUI (*e.g.* one can see in this figure the namespaces of the concepts in the concept hierarchy).

Tools Menu

In the "Tools" menu all additional plugins can be managed. Initially the list of available (additional) plugins is empty, one can only select to open the plugin management like shown in Figure A.19.

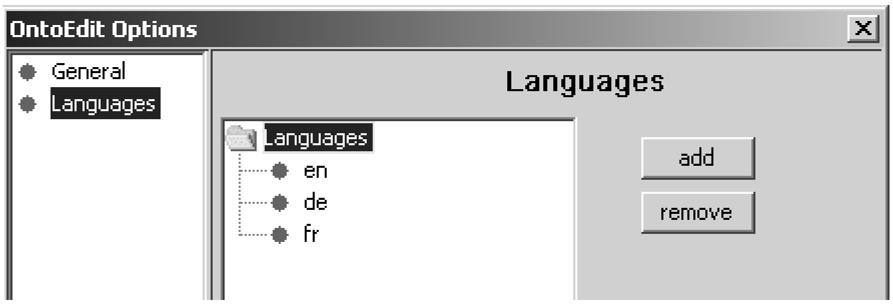


Figure A.18: Options for plugins

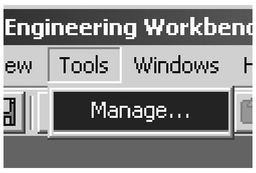


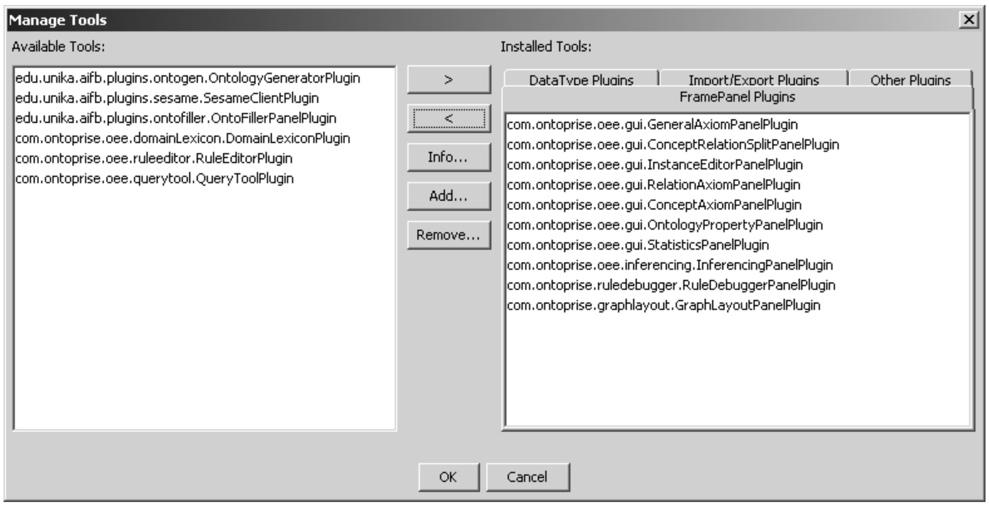
Figure A.19: Tools menu (i)

The plugin management allows for dynamically plugging in and out the plugins during runtime of OntoEdit (*cf.* Figure A.20). To make plugins available for plugging in, their class files need to be present in the classpath of OntoEdit. The complete package reference is needed to register a plugin with the "Add" button, *e.g.* `edu.unika.aifb.sesame.SesameClientPlugin` for the "SesameClientPlugin" plugin. By selecting plugins from the left part of the window and pressing the ">" button, one may register plugins. Deregistration works similar via the "<" button in combination with the right window.

After successfully registering plugins they are immediately available in OntoEdit as shown in Figure A.21. Plugins by default provide entries in the tools menu, but they might also provide buttons, *e.g.* the "OntologyGenerator" plugin provides"Generate ontology" as menu entry *and* as button, additional tabs or additional entries in the import and export menu (see above), to name but the typical possibilities.

Windows Menu

In the "Windows" menu one has the typical "Windows-like" features for managing multiple open ontology windows.


Manage Tools

Available Tools:

edu.unika.aifb.plugins.ontogen.OntologyGeneratorPlugin
edu.unika.aifb.plugins.sesame.SesameClientPlugin
edu.unika.aifb.plugins.ontofiller.OntoFillerPanelPlugin
com.ontoprise.oee.domainLexicon.DomainLexiconPlugin
com.ontoprise.oee.ruleeditor.RuleEditorPlugin
com.ontoprise.oee.querytool.QueryToolPlugin

Installed Tools:

DataType Plugins | Import/Export Plugins | Other Plugins | FramePanel Plugins

com.ontoprise.oee.gui.GeneralAxiomPanelPlugin
com.ontoprise.oee.gui.ConceptRelationSplitPanelPlugin
com.ontoprise.oee.gui.InstanceEditorPanelPlugin
com.ontoprise.oee.gui.RelationAxiomPanelPlugin
com.ontoprise.oee.gui.ConceptAxiomPanelPlugin
com.ontoprise.oee.gui.OntologyPropertyPanelPlugin
com.ontoprise.oee.gui.StatisticsPanelPlugin
com.ontoprise.oee.inferencing.InferencingPanelPlugin
com.ontoprise.ruledebugger.RuleDebuggerPanelPlugin
com.ontoprise.graphlayout.GraphLayoutPanelPlugin

Figure A.20: Plugin management during runtime

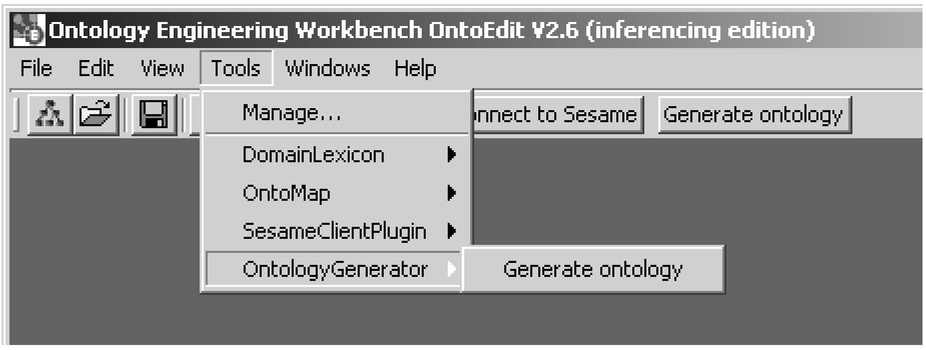


Figure A.21: Tools menu (ii)

Help Menu

The “Help” menu includes information about the version of OntoEdit, contact information and a user’s guide which is similar to this section.

B OntoEdit Inside

Overview

This appendix provides a detailed view on the implementation of OntoEdit by instantiating the OntoMat framework. It therefore extends the rather conceptually oriented description in Section 6.3. The appendix starts with a brief description of the history of OntoMat, then continues with a description of all interfaces common to all OntoMat plugins followed by a brief illustration of OntoEdit's core plugins and finally ends with the planned development steps of the OntoMat framework.

History & Facts

OntoMat was mainly developed by Siegfried Handschuh (Institute AIFB, University of Karlsruhe, Germany) and is publicly available as Open Source. OntoMat is developed purely in Java. The source code of the OntoMat plugin framework is available for download at Source Forge at `http://sourceforge.net/projects/ontomat`. Other tools based on this framework include the OntoMat–*Annotizer* (also mainly developed by Siegfried Handschuh), a tool that is designed for supporting semi-automatic annotation of heterogeneous data sources (Handschuh et al., 2001; 2002).

Interfaces

In order to build a plugin for the OntoMat framework, either directly the `OntoPlugin` interface must be implemented or one of its descendants like `OntoPluginServiceConsumer` or `OntoPluginServiceProvider`.

In order to simplify the implementation of a plugin there are specialized classes that already implemented these interfaces. *E.g.* so-called tab-widget plugins are user interface tabs that appear in the main OntoEdit window beside the system tabs such as the classes tab. A tab-widget plugin is simple to write, but it is less powerful.

OntoPlugin

The following methods of the `OntoPlugin` interface must be implemented by a plugin.

- `init`: This method is called after instantiation of the plugin. The plugin should perform any initializations here. The plugin should try to load its preferences via the framework.
- `canExit`: If any of the plugin returns false, the exit process of the framework is interrupted.
- `exit`: plugin should perform cleanup operations here. First the plugin's `canExit()` method is called from the framework. If it returns true, its `exit()` method is called. After that, the plugin is removed from the internal cache and its buttons and menu are removed from the GUI.
- `getPluginInfo`: The plugin can return here a short information message over itself, *e.g.* the copyright, the author and the intended purpose.
- `getPluginMenuName`: Should return a name for the plugin that can be displayed in a menu bar. Can return `null` when there should be no menu item for this plugin.
- `getPluginName`: Should return a user-friendly name for the plugin.
- `getToolbarActions`: plugins that require toolbar buttons should return the actions here. The buttons will appear in an own separator section.
- `getErrorMsg`: Should return an error message describing the status of the plugin. May return `null` if the status is ok.
- `getMenuActions`: plugins that require menu actions should return them here. The menu items will appear under the Tools | -getPluginMenuName- item.
- `getOptionsPane`: This method should return a subclass of `JComponent` that implements an option page for the plugin. The option pages will be displayed by the framework when users select the View | Options... menu item. The component should not extend the dimensions (350, 350) approximately. Using `JScrollPane` or better a `JTabbedPane` gives more space on the options pane. When the option component is about to be displayed, its `setVisible(true)` method is called. Again, if the component is to be hidden its `setVisible(false)` method is called. Any reading or writing of options should take place in that method.

OntoPluginServiceProvider

`OntoPluginServiceProvider` is the Java interface for an plugin that offer services in form of service classes. A plugin which implements this interface is hence called a **service-provider**. A plugin that implements the `OntoPluginServiceConsumer` interface is hence called a **service-consumer**.

A service-provider supplies on request an instance of the desired service class. The service class can be the plugin itself or a class on that the plugin refers. The service-provider may always return the same instance, or it may construct a new instance for each request. A service class specifies a interface protocol between the service-provider and the service-consumer.

The plugin framework has the function to act a as rendezvous between a service-consumer and a service-provider. The service-consumer may ask the framework to provide an instance of a "service", based upon a reference to a Java Class object that represent that service.

If such a service has been registered with the framework, then the service-provider associated with the service is asked to provide an instance of that service.

- `getCurrentServices`: This method is not used by the framework yet. In future this method will get the current service selectors for the specified service. A service selector is a service specific parameter, typical examples of which could include: a parameter to a constructor for the service implementation class, a value for a particular service's property, or a key into a map of existing implementations.
- `getService`: Invoked by the framework, this method requests an instance of a service.
- `getServiceClass`: Invoked by the framework, this method returns the service class.
- `releaseService`: Invoked by the framework, this method releases a reference to the specified service.

OntoPluginServiceConsumer

A plugin that implements the `OntoPluginServiceConsumer` interface registers its intent to be notified of new services.

- `serviceAvailable`: Invoked by the framework, the service-consumer is notified about a new service.

- `serviceRevoked`: Invoked by the framework, the service-consumer is informed about the deletion of the service.

Core Plugins of OntoEdit

We now illustrate the core plugins of OntoEdit. They shape OntoEdit as an Ontology Engineering Environment on top of the OntoMat plugin framework.

- The `GeneralPlugin` is an implementation of the `OntoPlugin` interface. It is always loaded during run-time and cannot be removed with the management console. This plugin represent the host framework. In the option menu it represents the settings for OntoEdit.
- The `OntologyServer` is a service-provider. It releases the access to a service class with the interface `IOntologyServer`. This service class contains the data model of the ontology. It supplies for the service-consumer methods for the access to the concepts, relations and instances of the ontology. Complex data types are avoided as parameters of the methods in order to achieve a good information hiding of the service design and therefore to support modularization.
- The `OntologyEditor` is a service-consumer. It uses the service, which is defined by the `IOntologyServer` interface. The `OntologyEditor` supports the construction and modification of ontologies, it visualizes elements of the ontology, *e.g.* the concepts, relations and instances.

Future Developments of OntoMat

The OntoMat plugin framework will be constantly improved. For the future following functions are planned:

- Plugin folder: User friendly installation of plugins. Automatic identification and loading of packed (jar)plugins from a certain folder.
- Internet based update: The framework will support an update mechanism. This will permit it to examine whether on the home page of a plugin a new version is present and to offer this new version to load automatically.
- Recursive plugin architecture: a plugin consist of plugins. For that it needs to be able to act as a plugin container like the framework.

C DAML+OIL features supported by OntoEdit

This appendix provides a detailed description of OntoEdit's support for importing and exporting DAML+OIL ontology language (*cf.* (Horrocks et al., 2001)). The description was posted to the "www-rdf-logic@w3.org" mailing list (*cf.* (Erdmann et al., 2001)).

Executive overview

According to the discussion about the capabilities of the different ontology editors, we would like to describe which language constructs of the DAML+OIL language are supported by OntoEdit 2.0 and which will be supported in future versions. Confer `http://www.ontoprise.de/com/start_downlo.htm` for additional information.

OntoEdit fully supports the following DAML+OIL constructs:

- `daml:Ontology`
- `daml:versionInfo`, `rdfs:comment`, `rdfs:label`
- `daml:Class`, `rdfs:Class`, `rdfs:subClassOf`
- `rdfs:Property`
- `daml:Restriction`
- `daml:cardinality`, `daml:minCardinality`, `daml:maxCardinality`
- `daml:onProperty`, `daml:toClass`
- `daml:inverseOf`, `daml:TransitiveProperty`
- `daml:disjointWith`

We are working on the following features to include in OntoEdit:

- `daml:imports`
- `rdfs:subPropertyOf`
- `daml:DataTypeProperty`, `daml:ObjectProperty`

Note: DAML+OIL allows (and encourages) the use of “anonymous classes” that are defined by their properties[1]. In general, OntoEdit does not support these implicitly defined classes and expects named classes instead of class-expressions. Thus OntoEdit is somehow restricted to a more OO-like usage of DAML+OIL. Nevertheless, certain information contained in anonymous classes are handled by OntoEdit (*cf.* property-restrictions).

Detailed description

The structure of the following list mirrors roughly the structure of the DAML+OIL reference description (`http://www.daml.org/2001/03/reference.html`).

Header
`<http://www.daml.org/2001/03/reference.html#Header>`

The DAML+OIL header `daml:Ontology` for ontology definitions and `daml:versionInfo` for metadata information about the ontology are fully supported.

Information from `daml:imports` is currently lost. This topic is on our mid-term to-do list, and already realized in OntoEdit’s basic datamodel. Thus only the GUI and import and export filters must be updated.

Objects and datatype values
`<http://www.daml.org/2001/03/reference.html#Object>`

OntoEdit supports instances of classes/concepts as well as atomic values. These atomic values are assumed to be “instances” of XML-Schema datatypes or of types that are derived from XML-Schema. Currently, XML-Schema definitions are not processed while importing DAML+OIL ontologies.

Class elements
`<http://www.daml.org/2001/03/reference.html#Class>`
and class expressions

[1]The DAML+OIL concepts “class” and “property” are similar to the concepts “concept” and “relation” in (Bozsak et al., 2002)

The basic DAML+OIL constructs for defining classes and class hierarchies (`daml:Class`, `rdfs:Class`, `rdfs:subClassOf`) are supported.

The `daml:disjointWith` property of classes will be supported in the next implementation of OntoEdit.

The `daml:disjointUnionOf` property and all properties assigning equivalence between classes, properties or instances are not supported (*e.g.* `daml:equivalentTo` or `daml:sameClassAs`).

Enumerations
`<htp://www.daml.org/2001/03/reference.html#Enumerated>`

The definition of a class that is defined by a closed list of its members (*i.e.* the `daml:oneOf` element) is not supported.

Property restrictions
`<http://www.daml.org/2001/03/reference.html#Restriction>`

Properties restrictions are handled by OntoEdit although they usually represent anonymous classes. If such an anonymous class is subclassed by a named class the value of the `daml:onProperty` property is assumed to represent a local property of the subclassing class and all properties of this property belong to the local property. The range of the property is assumed to be the single named class in `daml:toClass`.

The `daml:Restriction` is to define unqualified cardinality constraints are supported by OntoEdit. Their qualified parallels `daml:cardinalityQ` , `daml:minCardinalityQ` , and `daml:maxCardinalityQ` are not supported.

Boolean combination of class expressions
`<http://www.daml.org/2001/03/reference.html#Boolean>`

It is not planned to support the set operators (`daml:intersectionOf` , `daml:unionOf` , `daml:complementOf`) for classes.

Property elements
`<http://www.daml.org/2001/03/reference.html#Property>`

OntoEdit also supports both global properties and local properties that are restricted to single classes

A property may have at most one rdfs:domain (a named class), *cf.* `daml:toClass` above and at most one named class as its `rdfs:range`.

A distinction between `daml:ObjectProperty` and `daml:DatatypeProperty` is currently not made.

Modelling relation hierarchies with rdfs:subPropertyOf will be representable in future versions but will not be supported by the GUI, *i.e.* one might import/export rdfs:subPropertyOf statements but will not be able to model it explicitly.

Again, the equivalence properties of properties are not supported. The properties `daml:inverseOf` and `daml:TransitiveProperty` are supported, while uniqueness and unambiguousness are not.

Instances
`<http://www.daml.org/2001/03/reference.html#Instances>`

All legal RDF instances are handled as DAML+OIL instances by OntoEdit, irrespective of their actual serialized (XML/RDF) format. Again, the equivalence properties of instances are not supported.

Datatype Values
`<http://www.daml.org/2001/03/reference.html#Values>`

Atomic values are handled in the way as the DAML+OIL reference proposes. This is achieved by interpreting all XML-Schema datatypes as special built-in concepts.

`rdf:parseType=“daml:collection”`
`<http://www.daml.org/2001/03/reference.html#collection>`

The DAML+OIL collections and its RDF primitives are not supported by OntoEdit.

D Example Ontology

Overview

This appendix contains the ontology that is used as a running example in Parts II and III. It is represented in OXML (*cf.* (Ontoprise, 2002c)), the home storage format of OntoEdit.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--Ontology written by Oxml2Writer, (c) Ontoprise GmbH-->

<oxml:ontology
   xmlns:a="http://this.is.an/example"
   xmlns:oxml="http://schema.ontoprise.com/oxml/core/2.0"
   xmlns:oxsd="http://schema.ontoprise.com/datatypes"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
    "http://schema.ontoprise.com/oxml/core/2.0 oxml2.0.xsd"
   id="http://this.is.an/example#"
   rootConcept="a:DEFAULT_ROOT_CONCEPT"
   rootRelation="a:DEFAULT_ROOT_RELATION">

<oxml:concept id="a:DEFAULT_ROOT_CONCEPT"/>

<oxml:concept id="a:Software">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Software
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Software
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>
```

```
<oxml:concept id="a:OEE">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OEE
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#OEE
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:Tool"/>
</oxml:concept>

<oxml:concept id="a:OntoMat-Plugin">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoMat-Plugin
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#OntoMat-Plugin
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:Tool"/>
</oxml:concept>

<oxml:concept id="a:Methodology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Methodology
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Vorgehensweise
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:Architecture">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Architecture
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Architektur
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>
```

```
<oxml:concept id="a:Application">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Application
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Anwendung
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:Software"/>
</oxml:concept>

<oxml:concept id="a:Tool">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Tool
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Werkzeug
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:Software"/>
</oxml:concept>

<oxml:concept id="a:Person">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Person
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Person
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:Role">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Role
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Rolle
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:Document">
```

```
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Document
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Dokument
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:PhD_Thesis">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#PhD Thesis
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Doktorarbeit
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:Document"/>
</oxml:concept>

<oxml:concept id="a:Annotation_Editor">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Annotation Editor
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Annotationseditor
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:Tool"/>
</oxml:concept>

<oxml:concept id="a:Ontology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Ontology
        </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Ontologie
        </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:Case_Study">
  <oxml:externalRepresentation language="en">
```

```
    http://this.is.an/example#Case Study
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Fallstudie
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:Project">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Project
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Projekt
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:concept id="a:Topic">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Topic
  </oxml:externalRepresentation>
  <oxml:externalRepresentation language="de">
    http://this.is.an/example#Thema
  </oxml:externalRepresentation>
  <oxml:subConceptOf concept="a:DEFAULT_ROOT_CONCEPT"/>
</oxml:concept>

<oxml:relation id="a:DEFAULT_ROOT_RELATION"/>

<oxml:relation id="a:supports">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#support
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:supported_by"/>
</oxml:relation>

<oxml:relation id="a:supports"
    domain="a:Software"
    range="a:Methodology"
```

```
/>

<oxml:relation id="a:supported_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#supported by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:supports"/>
</oxml:relation>

<oxml:relation id="a:supported_by"
    domain="a:Methodology"
    range="a:Software"
/>

<oxml:relation id="a:implements">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#implements
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:implemented_by"/>
</oxml:relation>

<oxml:relation id="a:implements"
    domain="a:Software"
    range="a:Architecture"
/>

<oxml:relation id="a:implemented_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#implemented by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:implements"/>
</oxml:relation>

<oxml:relation id="a:implemented_by"
    domain="a:Architecture"
    range="a:Software"
/>
```

```
<oxml:relation id="a:has_part">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#has part
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:part_of"/>
</oxml:relation>

<oxml:relation id="a:has_part"
    domain="a:Software"
    range="a:Software"
/>

<oxml:relation id="a:has_part"
    domain="a:Project"
    range="a:Case_Study"
/>

<oxml:relation id="a:part_of">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#part of
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:has_part"/>
</oxml:relation>

<oxml:relation id="a:part_of"
    domain="a:Software"
    range="a:Software"
/>

<oxml:relation id="a:part_of"
    domain="a:Case_Study"
    range="a:Project"
/>

<oxml:relation id="a:version">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#version
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
```

```
</oxml:relation>

<oxml:relation id="a:version"
    domain="a:Software"
    range="xsd:STRING"
/>

<oxml:relation id="a:has_role">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#has role
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:carried_by"/>
</oxml:relation>

<oxml:relation id="a:has_role"
    domain="a:Person"
    range="a:Role"
/>

<oxml:relation id="a:developed_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#developed by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:develops"/>
</oxml:relation>

<oxml:relation id="a:developed_by"
    domain="a:Software"
    range="a:Person"
/>

<oxml:relation id="a:develops">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#develops
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:developed_by"/>
</oxml:relation>
```

```
<oxml:relation id="a:develops"
    domain="a:Person"
    range="a:Software"
/>

<oxml:relation id="a:authored_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#authored by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:authors"/>
</oxml:relation>

<oxml:relation id="a:authored_by"
    domain="a:Document"
    range="a:Person"
/>

<oxml:relation id="a:authors">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#authors
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:authored_by"/>
</oxml:relation>

<oxml:relation id="a:authors"
    domain="a:Person"
    range="a:Document"
/>

<oxml:relation id="a:has_title">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#has title
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
</oxml:relation>

<oxml:relation id="a:has_title"
    domain="a:Document"
    range="xsd:STRING"
```

```
/>

<oxml:relation id="a:invented_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#invented by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:invent"/>
</oxml:relation>

<oxml:relation id="a:invented_by"
    domain="a:Architecture"
    range="a:Person" />

<oxml:relation id="a:invent">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#invent
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:invented_by"/>
</oxml:relation>

<oxml:relation id="a:invent"
    domain="a:Person"
    range="a:Architecture"
/>

<oxml:relation id="a:carried_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#carried by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:has_role"/>
</oxml:relation>

<oxml:relation id="a:carried_by"
    domain="a:Role"
    range="a:Person"
/>

<oxml:relation id="a:used_to_build">
```

```
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#used to build
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
</oxml:relation>

<oxml:relation id="a:used_to_build"
    domain="a:Tool"
    range="a:Application"
/>

<oxml:relation id="a:used_to_engineer">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#used to engineer
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:engineered_with"/>
</oxml:relation>

<oxml:relation id="a:used_to_engineer"
    domain="a:OEE"
    range="a:Ontology"
/>

<oxml:relation id="a:engineered_with">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#engineered with
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:used_to_engineer"/>
</oxml:relation>

<oxml:relation id="a:engineered_with"
    domain="a:Ontology"
    range="a:OEE"
/>

<oxml:relation id="a:has_participant">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#has participant
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:participates_in"/>
</oxml:relation>

<oxml:relation id="a:has_participant"
    domain="a:Project"
    range="a:Person"
/>

<oxml:relation id="a:participates_in">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#participates in
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:has_participant"/>
</oxml:relation>

<oxml:relation id="a:participates_in"
    domain="a:Person"
    range="a:Project"
/>

<oxml:relation id="a:carries_out">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#carries out
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:carried_out_by"/>
</oxml:relation>

<oxml:relation id="a:carries_out"
    domain="a:Person"
    range="a:Case_Study"
/>

<oxml:relation id="a:carried_out_by">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#carried out by
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:inverseRelationOf relation="a:carries_out"/>
```

```
</oxml:relation>

<oxml:relation id="a:carried_out_by"
    domain="a:Case_Study"
    range="a:Person"
/>

<oxml:relation id="a:developed_in">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#developed in
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
</oxml:relation>

<oxml:relation id="a:developed_in"
    domain="a:Software"
    range="a:Project"
/>

<oxml:relation id="a:is_about">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#is about
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
</oxml:relation>

<oxml:relation id="a:is_about"
    domain="a:Document"
    range="a:Topic"
/>

<oxml:relation id="a:is_about"
    domain="a:Project"
    range="a:Topic"
/>

<oxml:relation id="a:is_about"
    domain="a:Case_Study"
    range="a:Topic"
/>
```

```
<oxml:relation id="a:is_about"
    domain="a:Ontology"
    range="a:Topic"
/>

<oxml:relation id="a:subtopic_of">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#subtopic of
  </oxml:externalRepresentation>
  <oxml:subRelationOf relation="a:DEFAULT_ROOT_RELATION"/>
  <oxml:algebraicProperty property="transitive"/>
</oxml:relation>

<oxml:relation id="a:subtopic_of"
    domain="a:Topic"
    range="a:Topic"
/>

<oxml:instance id="a:OntoEdit">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoEdit
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OEE"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Dirk_Wenke"/>
</oxml:instance>

<oxml:instance id="a:Mind2Onto">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Mind2Onto
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:York_Sure"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Markus_Zondler"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:supports"
```

```
    instance="a:On-To-Knowledge_Methodology"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
</oxml:instance>

<oxml:instance id="a:OntoAnalyzer">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoAnalyzer
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Dirk_Wenke"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:supports"
    instance="a:On-To-Knowledge_Methodology"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
</oxml:instance>

<oxml:instance id="a:OntoClean_Plugin">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoClean Plugin
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:York_Sure"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Stefan_Lenhart"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:supports"
    instance="a:On-To-Knowledge_Methodology"/>
  <oxml:hasRelation relation="a:supports"
    instance="a:OntoClean_Methodology"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
```

```
</oxml:instance>

<oxml:instance id="a:OntoFiller">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoFiller
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:York_Sure"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Heiko_Rudat"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:supports"
    instance="a:On-To-Knowledge_Methodology"/>
</oxml:instance>

<oxml:instance id="a:OntoGenerator">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoGenerator
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:York_Sure"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Stefan_Lenhart"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
</oxml:instance>

<oxml:instance id="a:OntoKick">
```

```
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoKick
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:York_Sure"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Claus_Boyens"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:supports"
    instance="a:On-To-Knowledge_Methodology"/>
</oxml:instance>

<oxml:instance id="a:Sesame_Plugin">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Sesame Plugin
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OntoMat-Plugin"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:York_Sure"/>
  <oxml:hasRelation relation="a:developed_by"
    instance="a:Stefan_Lenhart"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
  <oxml:hasRelation relation="a:part_of"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:supports"
    instance="a:On-To-Knowledge_Methodology"/>
</oxml:instance>

<oxml:instance id="a:Protege">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Protege
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OEE"/>
</oxml:instance>

<oxml:instance id="a:WebODE">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#WebODE
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:OEE"/>
</oxml:instance>

<oxml:instance id="a:SkiM">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#SkiM
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Application"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OTK_Architecture"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:On-To-Knowledge"/>
</oxml:instance>

<oxml:instance id="a:OntoWeb_Portal">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoWeb Portal
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Application"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:SEAL"/>
  <oxml:hasRelation relation="a:developed_in"
    instance="a:OntoWeb"/>
</oxml:instance>

<oxml:instance id="a:Dirk_Wenke">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Dirk Wenke
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
  <oxml:hasRelation relation="a:develops"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:has_role"
    instance="a:Developer"/>
```

```
</oxml:instance>

<oxml:instance id="a:Siegfried_Handschuh">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Siegfried Handschuh
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
</oxml:instance>

<oxml:instance id="a:York_Sure">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#York Sure
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
  <oxml:hasRelation relation="a:participates_in"
    instance="a:On-To-Knowledge"/>
  <oxml:hasRelation relation="a:participates_in"
    instance="a:OntoWeb"/>
</oxml:instance>

<oxml:instance id="a:Author">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Author
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Role"/>
</oxml:instance>

<oxml:instance id="a:Developer">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Developer
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Role"/>
</oxml:instance>

<oxml:instance id="a:This_Phd_Thesis">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#This Phd Thesis
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:PhD_Thesis"/>
  <oxml:hasRelation relation="a:authored_by"
    instance="a:York_Sure"/>
  <oxml:hasAttribute relation="a:has_title" range="xsd:STRING">
    <oxml:value>
        Methodology, Tools & Case Studies
        for Ontology-based
        Knowledge Management
    </oxml:value>
  </oxml:hasAttribute>
</oxml:instance>

<oxml:instance id="a:OntoMat">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoMat
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Architecture"/>
  <oxml:hasRelation relation="a:invented_by"
    instance="a:Siegfried_Handschuh"/>
  <oxml:hasRelation relation="a:implemented_by"
    instance="a:OntoEdit"/>
</oxml:instance>

<oxml:instance id="a:SEAL">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#SEAL
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Architecture"/>
</oxml:instance>

<oxml:instance id="a:OTK_Architecture">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OTK Architecture
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Architecture"/>
</oxml:instance>

<oxml:instance id="a:On-To-Knowledge_Methodology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#On-To-Knowledge Methodology
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Methodology"/>
</oxml:instance>
```

```
<oxml:instance id="a:OntoClean_Methodology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoClean Methodology
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Methodology"/>
</oxml:instance>

<oxml:instance id="a:Claus_Boyens">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Claus Boyens
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
</oxml:instance>

<oxml:instance id="a:Heiko_Rudat">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Heiko Rudat
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
</oxml:instance>

<oxml:instance id="a:Stefan_Lenhart">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Stefan Lenhart
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
</oxml:instance>

<oxml:instance id="a:Markus_Zondler">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Markus Zondler
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Person"/>
</oxml:instance>

<oxml:instance id="a:OntoMat-Annotizer">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoMat-Annotizer
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Annotation_Editor"/>
  <oxml:hasRelation relation="a:developed_by"
```

```
    instance="a:Siegfried_Handschuh"/>
  <oxml:hasRelation relation="a:implements"
    instance="a:OntoMat"/>
</oxml:instance>

<oxml:instance id="a:On-To-Knowledge">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#On-To-Knowledge
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Project"/>
  <oxml:hasRelation relation="a:has_part"
    instance="a:BT_Case_Study"/>
  <oxml:hasRelation relation="a:has_part"
    instance="a:EnerSearch_Case_Study"/>
  <oxml:hasRelation relation="a:has_part"
    instance="a:Swiss_Life_Case_Study"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Semantic_Web"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Knowledge_Management"/>
</oxml:instance>

<oxml:instance id="a:OntoWeb">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoWeb
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Project"/>
  <oxml:hasRelation relation="a:is_about" instance="a:Semantic_Web"/>
</oxml:instance>

<oxml:instance id="a:Ontology_Engineering">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Ontology Engineering
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Engineering"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Semantic_Web"/>
</oxml:instance>
```

```
<oxml:instance id="a:Knowledge_Engineering">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Knowledge Engineering
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Management"/>
</oxml:instance>

<oxml:instance id="a:Knowledge_Management">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Knowledge Management
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
</oxml:instance>

<oxml:instance id="a:Semantic_Web">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Semantic Web
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
</oxml:instance>

<oxml:instance id="a:Skills_Management">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Skills Management
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Management"/>
</oxml:instance>

<oxml:instance id="a:Virtual_Organization">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Virtual Organization
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Management"/>
</oxml:instance>
```

```
<oxml:instance id="a:Knowledge_Sharing">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Knowledge Sharing
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Management"/>
</oxml:instance>

<oxml:instance id="a:Community_of_Practice">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Community of Practice
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Management"/>
</oxml:instance>

<oxml:instance id="a:Knowledge_Portal">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Knowledge Portal
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Topic"/>
  <oxml:hasRelation relation="a:subtopic_of"
    instance="a:Knowledge_Management"/>
</oxml:instance>

<oxml:instance id="a:Swiss_Life_Case_Study">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Swiss Life Case Study
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Case_Study"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Skills_Management"/>
</oxml:instance>

<oxml:instance id="a:BT_Case_Study">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#BT Case Study
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Case_Study"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Knowledge_Sharing"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Community_of_Practice"/>
</oxml:instance>

<oxml:instance id="a:EnerSearch_Case_Study">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#EnerSearch Case Study
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Case_Study"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Virtual_Organization"/>
</oxml:instance>

<oxml:instance id="a:Swiss_Life_ontology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#Swiss Life ontology
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Ontology"/>
  <oxml:hasRelation relation="a:engineered_with"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Skills_Management"/>
</oxml:instance>

<oxml:instance id="a:EnerSeach_ontology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#EnerSeach ontology
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Ontology"/>
  <oxml:hasRelation relation="a:engineered_with"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Virtual_Organization"/>
</oxml:instance>

<oxml:instance id="a:BT_ontology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#BT ontology
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Ontology"/>
  <oxml:hasRelation relation="a:engineered_with"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Community_of_Practice"/>
</oxml:instance>

<oxml:instance id="a:OntoWeb_ontology">
  <oxml:externalRepresentation language="en">
    http://this.is.an/example#OntoWeb ontology
  </oxml:externalRepresentation>
  <oxml:instanceOf concept="a:Ontology"/>
  <oxml:hasRelation relation="a:engineered_with"
    instance="a:OntoEdit"/>
  <oxml:hasRelation relation="a:is_about"
    instance="a:Semantic_Web"/>
</oxml:instance>

<oxml:axiom id="a:flogic_axiom1" enabled="true">
  <oxml:text language="flogic">
    FORALL P ( P[#&quot;\&quot;has role\&quot;&quot;->>&quot;
        http://this.is.an/example&quot;#Author] )
    &lt;- ( EXISTS D(P[&quot;http://this.is.an/example
        &quot;#authors->>D]) ).
  </oxml:text>
</oxml:axiom>

<oxml:axiom id="a:flogic_axiom2" enabled="true">
  <oxml:text language="flogic">
    FORALL P ( P[#&quot;\&quot;has role\&quot;&quot;->>&quot;
        http://this.is.an/example&quot;#Developer] )
    &lt;- ( EXISTS S(P[&quot;http://this.is.an/example
        &quot;#develops->>S]) ).
  </oxml:text>
</oxml:axiom>

<oxml:module id="DomainLexiconModule1040341981773"
    title="domain lexicon module" version="1.0"
    processor="com.ontoprise.oee.domainLexicon.DomainLexiconPlugin"
    xmlns:a="http://this.is.an/example"
    xmlns:domlex="http://schema.ontoprise.com/oxml/extensions/
    domainLexicon">
</oxml:module>

<oxml:module id="OntoMapModule1040341981783"
    title="ontoMap^module" version="1.0"
    processor="com.ontoprise.oee.ontomap.OntoMapPlugin"
    xmlns:a="http://this.is.an/example"
    xmlns:ontomap="http://schema.ontoprise.com/oxml/extensions/
    ontomap">
</oxml:module>

<oxml:module id="Mind2Onto"
    processor="edu.unika.aifb.plugins.ontoskin.OntoskinPlugin"
    xmlns:a="http://this.is.an/example"
    xmlns:mind2onto="http://schema.ontoprise.com/oxml/extension/
    mind2onto/0.1 mind2onto_schema.xsd">

<mind2onto:skin concept="a:OntoMat-Plugin">
    <mind2onto:code id="9"/>
</mind2onto:skin>

<mind2onto:skin concept="a:OEE">
    <mind2onto:type>Arial</mind2onto:type>
    <mind2onto:size>10</mind2onto:size>
    <mind2onto:code id="7"/>
</mind2onto:skin>

<mind2onto:skin concept="a:Methodology">
    <mind2onto:type>Arial Black</mind2onto:type>
    <mind2onto:size>14</mind2onto:size>
    <mind2onto:highlightColor blue="51" green="51" red="255"/>
    <mind2onto:code id="5"/>
</mind2onto:skin>

<mind2onto:skin concept="a:Tool">
    <mind2onto:code id="5"/>
</mind2onto:skin>

</oxml:module>

</oxml:ontology>
```

Bibliography

Abecker, A., Bernardi, A., Hinkelmann, K., Kuehn, O., & Sintek, M. (1998). Toward a technology for organizational memories. *IEEE Intelligent Systems*, 13(3):40–48.

Abecker, A., Decker, S., Hinkelmann, K., & Reimer, U. (Eds.) (1997). *Proceedings of the International Workshop on Knowledge-Based Systems for Knowledge Management in Enterprises.* 21. Deutsche Jahrestagung “Künstliche Intelligenz” (KI-97), Freiburg, September, 1997, available at http://www.dfki.uni-kl.de/km/ws-ki-97.html.

Abiteboul, S. & Vercoustre, A.-M. (Eds.) (1999). *Proceedings of the Third European Conference on Digital Libraries (ECDL-99): Research and Advanced Technology for Digital Libraries*, volume 1696 of *Lecture Notes in Computer Science (LNCS)*, Paris, France. Springer.

AIdministrator (2003a). Sesame SourceForge. http://sourceforge.net/projects/sesame.

AIdministrator (2003b). Sesame website. http://sesame.aidministrator.nl/.

Albrecht, F. (1993). *Strategisches Wissensmanagement der Unternehmensressource Wissen.* Verlag Peter Lang, Frankfurt am Main.

Amann, B. & Fundulaki, I. (1999). Integrating ontologies and thesauri to build RDF schemas. In (Abiteboul & Vercoustre, 1999).

Anderson, C. R., Levy, A. Y., & Weld, D. S. (1999). Declarative web site management with tiramisu. In *ACM SIGMOD Workshop on the Web and Databases (WebDB99)*, pages 19–24.

Angele, J., Schnurr, H.-P., Staab, S., & Studer, R. (2000). The times they are a-changin' – the Corporate History AnalyzeR. In (Reimer & Mahling, 2000), pages 1:1–1:11. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-34/.

Angele, J. (1993). *Operationalisierung des Modells der Expertise mit KARL.* Number 53 in DISKI. infix, St. Augustin. PhD Thesis.

Arpírez, J. C., Corcho, O., Fernández-López, M., & Gómez-Pérez, 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.*

Aussenac-Gilles, N., Biébow, B., & Szulman, S. (2002). Modelling the travelling domain from a NLP description with TERMINAE. In (Sure & Angele, 2002), pages 112–128. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Becerra-Fernandez, I. (2000). The role of artificial intelligence technologies in the implementation of people-finder knowledge management systems. In Staab, S. & O'Leary, D. (Eds.), *Bringing Knowledge to Business Processes. Workshop in the AAAI Spring Symposium Series. Stanford, March 20-22, 2000*, Menlo Park, CA. AAAI.

Bechhofer, S., Horrocks, I., Goble, C., & Stevens, R. (2001). OilEd: A reason-able ontology editor for the semantic web. In *KI-2001: Advances in Artificial Intelligence*, LNAI 2174, pages 396–408. Springer.

Bechhofer, S. (2002). Travelling domain experiment: Preliminary results for OilEd. In (Sure & Angele, 2002), pages 80–82. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Benjamins, V. R., Fensel, D., Decker, S., & Gómez-Pérez, A. (1999). $(KA)^2$: Building ontologies for the internet. *International Journal of Human-Computer Studies (IJHCS)*, 51(1):687–712.

Benjamins, V. R. & Fensel, D. (1998). Community is knowledge! $(KA)^2$. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management (KAW '98), Banff, Canada.*

Berendt, B., Hotho, A., & Stumme, G. (Eds.) (2002a). *Proceedings of the Second Workshop on Semantic Web Mining.* Held in conjunction with (Elomaa et al., 2002a) and (Elomaa et al., 2002b); Online available at http://km.aifb.uni-karlsruhe.de/semwebmine2002/.

Berendt, B., Hotho, A., & Stumme, G. (2002b). Towards semantic web mining. In (Horrocks & Hendler, 2002), pages 264–278.

Bernaras, A., Laresgoiti, I., & Corera, J. (1996). Building and reusing ontologies for electrical network applications. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'96).*

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.

Berners-Lee, T. (1993). Naming and addressing: URIs, URLs, ... W3C Overview. available at http://www.w3.org/Addressing/.

Berners-Lee, T. (1998). Cool URIs don't change. W3C Style. available at http://www.w3.org/Provider/Style/URI.html.

Bernstein, P. A., Hadzilacos, V., & Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Reading, MA.

Biron, P. V. & Malhotra, A. (2001). XML Schema Part 2: Datatypes. W3C Recommendation 02 May 2001. available at http://www.w3.org/TR/xmlschema-2/.

Boehm, B. (1984). Verifying and validating software requirements and design specifications. *IEEE Software*, 1(1):75–88.

Borst, W. N. & Akkermans, J. M. (1997). Engineering ontologies. *International Journal of Human-Computer Studies*, 46(2/3):365–406.

Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., & Zacharias, V. (2002). KAON – towards a large scale semantic web. In Bauknecht, K., Tjoa, A. M., & Quirchmayr, G. (Eds.), *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web 2002)*, volume 2455 of *LNCS*, pages 304–313, Aix-en-Provence, France. Springer.

Bray, T., Hollander, D., & Layman, A. (1999). Namespaces in XML. W3C Recommendation 14 January 1999. available at http://www.w3.org/TR/REC-xml-names/.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (2000). Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation 06 October 2000. available at http://www.w3.org/TR/REC-xml.

Brickley, D. & Guha, R. V. (2002). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 12 November 2002. available at 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 8, AIdministrator Nederland b.v.

Broekstra, J., Kampman, A., & van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF Schema. In (Horrocks & Hendler, 2002), pages 54–68.

Broekstra, J. & Kampman, A. (2000). Query language definition. On-To-Knowledge deliverable 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 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 *Proceedings of the 10th International World Wide Web Conference (WWW10)*, Hong Kong.

Brown, M., Fernandéz-López, M., Gómez-Pérez, A., & Léger, A. (2002). Business scenarios. OntoWeb deliverable 1.2, Semantic Edge GmbH.

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

Ceri, S., Fraternali, P., & Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. In *Proceedings of the 9th World Wide Web Conference (WWW9)*.

Ceri, S., Fraternali, P., & Paraboschi, S. (1999). Data-driven one-to-one web site generation for data-intensive applications. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 615–626.

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., Fernández-López, M., & Gómez-Pérez, A. (2002). Evaluation experiment for the editor of the WebODE ontology workbench. In (Sure & Angele, 2002), pages 129–134. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Corcho, O. & Gómez-Pérez, A. (2000). A roadmap to ontology specification languages. In (Dieng & Corby, 2002), pages 80–96.

Crampes, M. & Ranwez, S. (2000). Ontology-supported and ontology-driven conceptual navigation on the world wide web. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia*, pages 191–199. ACM Press.

Dahlgren, K. (1995). A linguistic ontology. *International Journal of Human–Computer Studies*, 43(5/6):809–818.

Davenport, T. H., Jarvenpaa, S. L., & Beers, M. C. (1996). Improving knowledge work processes. *Sloan Management Review*, 37(4):53–65.

Davenport, T. H. & Prusak, L. (1998). *Working Knowledge – How organisations manage what they know.* Havard Business School Press, Boston, Massachusetts.

Davenport, T. H. (1996). Some principles of knowledge management. Technical report, Graduate School of Business, University of Texas at Austin, Strategy and Business.

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

Davies, J., Duke, A., & Sure, Y. (2003). OntoShare – Evaluation of an ontology based knowledge sharing system. Submitted 2003.

Davies, J., Fensel, D., & van Harmelen, F. (Eds.) (2002b). *On-To-Knowledge: Semantic Web enabled Knowledge Management.* J. Wiley and Sons.

Davies, N. J. (2000). *Supporting Virtual Communities of Practice*, pages 199–212. In (Roy, 2000).

DC (2003). Dublin Core Metadata Initiative. http://dublincore.org/.

Decker, S., Daniel, M., Erdmann, M., & Studer, R. (1997). An enterprise reference scheme for integrating model based knowledge engineering and enterprise modeling. In (Plaza & Benjamins, 1997).

Decker, S., Erdmann, M., Fensel, D., & Studer, R. (1999). *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*, pages 351–369. In (Meersman et al., 1999).

Decker, S. (2002). *Semantic Web Methods for Knowledge Management.* PhD thesis, Institute AIFB, University of Karlsruhe.

Demey, J., Jarrar, M., & Meersman, R. (2002). A conceptual markup language that supports interoperability between business rules modeling systems. In (Meersman et al., 2002), pages 19–35.

Denny, M. (2002). Ontology editor survey results (table). available at http://xml.com/2002/11/06/Ontology_Editor_Survey.html.

De Raedt, L. & Flach, P. (Eds.) (2001). *Proceedings of the 12th European Conference on Machine Learning (ECML 2001)*, volume 2167 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag.

De Raedt, L. & Siebes, A. (Eds.) (2001). *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001)*, volume 2168 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag.

Dieng-Kuntz, R. & Matta, N. (Eds.) (2002). *Knowledge Management and Organizational Memories*. Kluwer Academic Publishers, Boston, Dordrecht, London.

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.

Dieng, R. & Corby, O. (Eds.) (2002). *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management: Methods, Models, and Tools (EKAW 2000)*, volume 1937 of *Lecture Notes in Artificial Intelligence (LNAI)*, Juan-les-Pins, France. Springer.

Dittmann, L., Peters, M. L., & Zelewski, S. (2003). Mitarbeitermotivation und Kompetenzmanagementsysteme. In (Sure & Schnurr, 2003). To appear 2003.

Domingue, J. (1998). Tadzebao and WebOnto: Discussing, browsing, and editing ontologies on the web. In *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, April 18th-23rd. Banff, Canada.*

Drucker, P. A. (1993). *A Post Capitalist Society*. HarperCollins, New York.

Duineveld, A. J., Stoter, R., Weiden, M. R., Kenepa, B., & Benjamins, V. R. (2000). Wondertools? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 6(52):1111–1133.

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

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

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

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

Eason, K. (1988). *Information Technology and Organisational Change*. Taylor & Francis, London.

Edvinson, L. & Malone, M. (1997). *Intellectual capital. Realizing your company's true value by finding its hidden brainpower.* Harper, New York.

Elbert, S. (2001). Einführung eines Management-Support-Systems zum effektiven Skill-Management bei Bertelsmann mediaSystems. In (Schnurr et al., 2001), pages 129–143.

Ellis, C. A., Gibbs, S. J., & Rein, G. L. (1991). Groupware, some issues and experiences. *Communications of the ACM*, 34(1).

Elomaa, T., Mannila, H., & Toivonen, H. (Eds.) (2002a). *Proceedings of the 13th European Conference on Machine Learning (ECML 2002)*, volume 2430 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag.

Elomaa, T., Mannila, H., & Toivonen, H. (Eds.) (2002b). *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002)*, volume 2431 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag.

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

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

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

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

Erdmann, M., Maedche, A., Schnurr, H.-P., & Staab, S. (2000). From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *Proceedings of the COLING-2000 Workshop on Semantic Annotation and Intelligent Content*, Centre Universitaire, Luxembourg.

Erdmann, M. & Studer, R. (2001). How to structure and access XML documents with ontologies. *Data and Knowledge Engineering*, 36(3):317–335.

Erdmann, M., Sure, Y., & Wenke, D. (2001). OntoEdit: Which DAML+OIL features are/will be supported? Posted on 2001-12-12 to the "www-rdf-logic@w3.org" mailing list. available at http://lists.w3.org/Archives/Public/www-rdf-logic/2001Dec/0018.html.

Erdmann, M. (2001). *Ontologien zur konzeptuellen Modellierung der Semantik von XML*. Books on Demand. PhD Thesis.

Farquhar, A., Fickas, R., & Rice, J. (1996). The Ontolingua Server: A tool for collaborative ontology construction. In *Proceedings of the 10th Banff Knowledge Acquisition for KnowledgeBased System Workshop (KAW'95)*, Banff, Canada.

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.) (2003). *Spinning the Semantic Web*. MIT Press.

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. (2000b). Ontology Language Version 1. On-To-Knowledge deliverable 1, Vrije Universiteit Amsterdam.

Fensel, D., Horrocks, I., van Harmelen, F., Decker, S., Erdmann, M., & Klein, M. (2000c). OIL in a nutshell. In (Dieng & Corby, 2002), pages 1–16.

Fensel, D., van Harmelen, F., Ding, Y., Klein, M., Akkermans, H., Broekstra, J., Kampman, A., van der Meer, J., Studer, R., Sure, Y., Davies, J., Duke, A., Engels, R., Iosif, V., Kiryakov, A., Lau, T., Reimer, U., & Horrocks, I. (2002). Final project report. On-To-Knowledge deliverable, 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 0, Vrije Universiteit Amsterdam.

Fensel, D. (1995). *The Knowledge Acquisition and Representation Language KARL*. Kluwer Academic Publisher, Boston.

Fensel, D. (2000). *Problem Solving Methods: Understanding, Description, Development, and Reuse*, volume 1791 of *Lecture Notes in Computer Science (LNCS)*. Springer.

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 2, Vrije Universiteit Amsterdam.

Fernandéz-López, M., Gómez-Pérez, A., Euzenat, J., Gangemi, A., Kalfoglou, Y., Pisanelli, D. M., Schorlemmer, M., Steve, G., Stojanovic, L., Stumme, G., & Sure, Y. (2002). A survey on methodologies for developing, maintaining, integrating, evaluating and reengineering ontologies. OntoWeb deliverable 1.4, Universidad Politecnia de Madrid.

Fernández-López, M., Gómez-Pérez, A., Sierra, J. P., & Sierra, A. P. (1999). Building a chemical ontology using Methontology and the Ontology Design Environment. *Intelligent Systems*, 14(1).

Fernández-López, M. (1999). Overview of methodologies for building ontologies. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*. CEUR Publications.

Fernandez, M. F., Florescu, D., Levy, A. Y., & Suciu, D. (2000). Declarative specification of web sites with Strudel. *VLDB Journal*, 9(1):38–55.

Fillies, C. & Sure, Y. (2002). On visualizing the Semantic Web in MS Office. In *6th International Conference on Information Visualisation (IV02)*, London, England.

Fillies, C. (2002). Evaluation experiment for ontology editors: SemTalk. In (Sure & Angele, 2002), pages 108–111. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

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

Fraternali, P. & Paolini, P. (1998). A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. In *EDBT 1998*, pages 421–435.

Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, NF 100:25–50. This article can also be found together with four others from G. Frege in (Frege, 1994).

Frege, G. (1994). *Funktion, Begriff, Bedeutung. Fünf logische Studien.* Kleine Vandenhoeck-Reihe. Vandenhoeck & Ruprecht, Göttingen.

Frohn, J., Himmeröder, R., Kandzia, P., & Schlepphorst, C. (1996). How to write F–Logic programs in FLORID. A tutorial for the database language F–Logic. Technical report, Institut fr̈ Informatik der Universität Freiburg. Version 1.0.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., & Schneider, L. (2002a). Sweetening ontologies with DOLCE. In (Gómez-Pérez & Benjamins, 2002), pages 166–181.

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

Gangemi, A. (2002). Travelling domain experiment: Results for Loom. In (Sure & Angele, 2002), pages 93–98. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Gil, Y., Gennari, J., & Porter, B. (Eds.) (2003). *Second International Conference on Knowledge Capture (K-CAP 2003)*. October 2003, Florida, USA; to appear 2003.

Gil, Y., Musen, M., & Shavlik, J. (Eds.) (2001). *Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001)*, New York, NY, USA. ACM Press.

Gluschko, R. J., Tenenebaum, J. M., & Meltzer, B. (1999). An XML framework for agent-based E-commerce. *Communications of the ACM*, 42(3):106–114.

Goble, C., Bechhofer, S., Carr, L., de Roure, D., & Hall, W. (2001). Conceptual open hypermedia = the semantic web? In *Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001, Hongkong, China, May 1, 2001*. CEUR Workshop Proceedings.

Gómez-Pérez, A., Angele, J., Fernandéz-López, M., Christophides, V., Stutt, A., Sure, Y., et al. (2002a). A survey on ontology tools. OntoWeb deliverable 1.3, Universidad Politecnia de Madrid.

Gómez-Pérez, A. & Benjamins, V. R. (Eds.) (2002). *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (EKAW 2002)*, volume 2473 of *Lecture Notes in Artificial Intelligence (LNAI)*, Siguenza, Spain. Springer.

Gómez-Pérez, A., Fernandéz-López, M., Corcho, O., Ahn, T. T., Aussenac-Gilles, N., Bernardos, S., Christophides, V., Corby, O., Crowther, P., Ding, Y., Engels, R., Esteban, M., Gandon, F., Kalfoglou, Y., Karvounarakis, G., Lama, M., López, A., Lozano, A., Magkanaraki, A., Manzano, D., Motta, E., Noy, N., Plexousakis, D., Ramos, J. A., & Sure, Y. (2002b). Technical roadmap. OntoWeb deliverable 1.1.2, Universidad Politecnia de Madrid.

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

Gómez-Pérez, A. (2003). *Ontology Evaluation*. In (Staab & Studer, 2003). To appear 2003.

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.

Gray, J. & Reuter, A. (1993). *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufman Publishers, Inc., San Francisco, CA.

Grosso, E., Eriksson, H., Fergerson, R. W., Tu, S. W., & Musen, M. M. (1999). Knowledge modeling at the millennium: the design and evolution of PROTEGE-2000. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW-99)*, Banff, Canada.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Gruber, T. R. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928.

Grueninger, M. & Fox, M. (1994). The role of competency questions in enterprise engineering. In *IFIP WG 5.7, Workshop Benchmarking. Theory and Practice*, Trondheim/Norway.

Guarino, N., Masolo, C., & Vetere, G. (1999). OntoSeek: Content-based access to the web. *IEEE Intelligent Systems*, 14(3).

Guarino, N. & Welty, C. (2000a). A formal ontology of properties. In (Dieng & Corby, 2002), pages 97–112.

Guarino, N. & Welty, C. (2000b). A formal ontology of properties. Technical report, LADSEB/CNR Technical Report 01/2000. available at http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html.

Guarino, N. & Welty, C. (2000c). Identity, unity, and individuality: Towards a formal toolkit for ontological analysis. *Proceedings of the European Conference on Artificial Intelligence (ECAI-2000), Berlin, 2000.*

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

Guarino, N. (1997). Understanding, building and using ontologies. *International Journal of Human and Computer Studies*, 46(2/3):293–310.

Guarino, N. (1998a). Formal ontology and information systems. In (Guarino, 1998b).

Guarino, N. (Ed.) (1998b). *Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS)*, volume 46 of *Frontiers in Artificial Intelligence and Applications*, Trento, Italy. IOS-Press.

Halpin, T. (2001). *Information Modelling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan-Kaufmann.

Handschuh, S., Staab, S., & Ciravegna, F. (2002). S-CREAM — Semi-automatic CREAtion of Metadata. In (Gómez-Pérez & Benjamins, 2002), pages 358–372.

Handschuh, S., Staab, S., & Maedche, A. (2001). CREAM – creating relational metadata with a component-based, ontology-driven annotation framework. In *Proceedings of the First International Conference on Knowledge Capture (K-Cap 2001)*, Victoria, B.C., Canada.

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

Hefke, M. & Trunko, R. (2002). A methodological basis for bringing knowledge management to real-world environments. In (Gómez-Pérez & Benjamins, 2002), pages 565–570.

Heflin, J. & Hendler, J. (2000). Searching the web with SHOE. In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop*, pages 35–40, Menlo Park, CA. AAAI Press. Technical Report WS-00-01.

Holsapple, C. W. (Ed.) (2003a). *Handbook on Knowledge Management 1 – Knowledge Matters*. International Handbooks on Information Systems. Springer, Berlin, Heidelberg, New York.

Holsapple, C. W. (Ed.) (2003b). *Handbook on Knowledge Management 2 – Knowledge Directions*. International Handbooks on Information Systems. Springer, Berlin, Heidelberg, New York.

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

Horrocks, I., van Harmelen, F., Patel-Schneider, P., Berners-Lee, T., Brickley, D., Connolly, D., Dean, M., Decker, S., Fensel, D., Fikes, R., Hayes, P., Heflin, J., Hendler, J., Lassila, O., McGuinness, D., & Stein, L. A. (2001). DAML+OIL (March 2001). Joint Committee, http://www.daml.org/2001/03/daml+oil-index.

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

Hou, C. J., Noy, N. F., & Musen, M. (2002). *A Template-based Approach Toward Acquisition of Logical Sentences*, pages 77–89. In (Musen et al., 2002).

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

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

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

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

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley, Reading, MA.

Jacobson, I. (1998). *Object-oriented Software Engineering*. Addison-Wesley, Reading, MA.

Jacques, R. (1996). *Manufacturing the employee – Management Knowledge from the 19th to 21st Centuries*. SAGE Publications, London, Thousand Oaks, New Delhi.

Jarrar, M. & Meersman, R. (2002). Formal ontology engineering in the DOGMA approach. In (Meersman et al., 2002), pages 1238–1254.

Jasper, R. & Uschold, M. (1999). A framework for understanding and classifying ontology applications. In (KAW, 1999). available at http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html.

Karagiannis, D. & Reimer, U. (Eds.) (2002). *Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management (PAKM2002)*, volume 2569 of *Lecture Notes in Artificial Intelligence (LNAI)*, Vienna, Austria. Springer.

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.

Kashyap, V. (1999). Design and creation of ontologies for environmental information retrieval. In (KAW, 1999). available at http://sern.ucalgary.ca/KSI/KAW/KAW99/papers/Kashyap1/kashyap.pdf.

KAW (1999). *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW-99)*, Banff, Canada. available at http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html.

Kay, A. S. (2003). *The Curious Success of Knowledge Management*, pages 679–687. In (Holsapple, 2003b).

Kesseler, M. (1996). A schema based approach to HTML authoring. *World Wide Web Journal*, 96(1).

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

Kifer, M. & Lozinskii, E. (1986). A framework for an efficient implementation of deductive databases. In *Proceedings of the 6th Advanced Database Symposium*, pages 109–116, Tokyo.

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

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

Klein, M., Fensel, D., Kiryakov, A., & Ognyanov, D. (2002a). Ontoview: Comparing and versioning ontologies. In *Collected Posters of ISWC 2002, cf. (Horrocks & Hendler, 2002).*

Klein, M., Fensel, D., Kiryakov, A., & Ognynanov, D. (2002b). Ontology versioning and change detection on the web. In (Gómez-Pérez & Benjamins, 2002), pages 197–212.

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

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

Kuehn, O. & Abecker, A. (1997). Corporate memories for knowledge memories in industrial practice: Prospects and challenges. *Journal of Universal Computer Science*, 3(8).

Labrou, Y. & Finin, T. W. (1999). Yahoo! as an ontology: Using Yahoo! categories to describe documents. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management*, pages 180–187, Kansas City, Missouri. ACM Press.

Landes, D. (1995). *Die Entwurfsphase in MIKE – Methode und Beschreibungssprache*. Number 84 in DISKI. infix, St. Augustin. PhD Thesis.

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

Lau, T. & Sure, Y. (2002). Introducing ontology-based skills management at a large insurance company. In *Proceedings of the Modellierung 2002*, pages 123–134, Tutzing, Germany.

Léger, A., Akkermans, H., Brown, M., Bouladoux, J.-M., Dieng, R., Ding, Y., Gómez-Pérez, A., Handschuh, S., Hegarty, A., Persidis, A., Studer, R., Sure, Y., Tamma, V., & Trousse, B. (2002a). Successful scenarios for ontology-based applications. OntoWeb deliverable 2.1, France Télécom R&D.

Léger, A., Bouillon, Y., Bryan, M., Dieng, R., Ding, Y., Fernandéz-López, M., Gómez-Pérez, A., Ecoublet, P., Persidis, A., & Sure, Y. (2002b). Best practices and guidelines. OntoWeb deliverable 2.2, France Télécom R&D.

Lei, Y., Motta, E., & Domingue, J. (2002). An ontology-driven approach to web site generation and maintenance. In (Gómez-Pérez & Benjamins, 2002), pages 219–234.

Lenat, D. B. & Guha, R. V. (1990). *Building large knowledge-based systems. Representation and inference in the CYC project*. Addison-Wesley, Reading, Massachusetts.

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of ACM*, 38(11):33–38.

Liao, M., Hinkelmann, K., Abecker, A., & Sintek, M. (1999). A competence knowledge base system as part of the organizational memory. In (Puppe, 1999), pages 125–137.

Maedche, A., Motik, B., Silva, N., & Volz, R. (2002a). MAFRA – a MApping FRAmework for distributed ontologies. In (Gómez-Pérez & Benjamins, 2002), pages 235–250.

Maedche, A., Staab, S., Stoijanovic, N., Studer, R., & Sure, Y. (2003). *SEmantic portAL – The SEAL approach*, chapter 11, pages 461–518. In (Fensel et al., 2003).

Maedche, A., Staab, S., Studer, R., Sure, Y., & Volz, R. (2002b). SEAL – tying up information integration and web site management by ontologies. *IEEE Computer Society Data Engineering Bulletin, Special Issue on Organizing and Discovering the Semantic Web*, 25(1):10–17.

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

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

Maedche, A. (2002b). Semantikbasiertes Wissensmanagement – Eine Anwendung in Human Resources. *Karlsruher Transfer*, Sommersemester 2002(07/01).

Majer, B., Studer, R., Sure, Y., & Volz, R. (2002). Web portal: Complete ontology and portal. OntoWeb deliverable 6.3, Institute AIFB, University of Karlsruhe & StarLAB, Vrije Universiteit Brussels.

McCarthy, J. (1959). Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London. Her Majesty's Office.

McCarthy, J. (1989). Artificial intelligence, logic and formalizing common sense. In Thomason, R. (Ed.), *Philosophical Logic and Artificial Intelligence*. Kluwer Academic, Dordrecht.

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

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

McKeen, J. D. & Staples, D. S. (2003). *Knowledge Managers: Who They Are and What They Do*, pages 21–41. In (Holsapple, 2003a).

Mecca, G., Merialdo, P., Atzeni, P., & Crescenzi, V. (1999). The (Short) Araneus Guide to Web-Site Development. In *Second International Workshop on the Web and Databases (WebDB'99) in conjunction with SIGMOD'99*.

Meersman, R., Tari, Z., et al. (Eds.) (2002). *Proceedings of the Confederated International Conferences: On the Move to Meaningful Internet Systems (CoopIS, DOA, and ODBASE 2002)*, volume 2519 of *Lecture Notes in Computer Science (LNCS)*, University of California, Irvine, USA. Springer.

Meersman, R., Tari, Z., & Stevens, S. (Eds.) (1999). *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic Publisher.

Meersman, R. (1999). Semantic ontology tools in information systems design. In (Raś & Skowron, 1999).

Mena, E., Kashyap, V., Illarramendi, A., & Sheth, A. (1998). Domain specific ontologies for semantic information brokering on the global information infrastructure. In (Guarino, 1998b).

Miller, G. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Minor, M. & Staab, S. (Eds.) (2002). *Proceedings of the 1st Workshop on Experience Management – Sharing Experiences about the Sharing of Experience*, volume P–10 of *Lecture Notes in Informatics (LNI)*, Berlin, Germany. Gesellschaft für Informatik.

Moench, E. (2003). SemanticMiner: Ein integratives Ontologie-basiertes Knowledge Retrieval System. In (Sure & Schnurr, 2003). To appear 2003.

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

Motik, B., Maedche, A., & Volz, R. (2002). A conceptual modeling approach for semantics–driven enterprise applications. In (Meersman et al., 2002), pages 1082–1099.

Musen, M., Neumann, B., & Studer, R. (Eds.) (2002). *Intelligent Information Processing*. Kluwer Academic Publishers, Boston, Dordrecht, London.

Neches, R., Fikes, R. E., Finin, T., Gruber, T. R., Senator, T., & Swartout, W. R. (1991). Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56.

Nejdl, W., Siberski, W., Simon, B., & Tane, J. (2002a). Towards a modification exchange language for distributed RDF repositories. In (Horrocks & Hendler, 2002), pages 236–249.

Nejdl, W., Wolf, B., QuLearning, C., Decker, S., Naeve, A., Nilsson, M., & Palmér, M. (2002b). EDUTELLA: A P2P networking infrastructure based on RDF. In *Proceedings of the Eleventh International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA. available at http://www2002.org/CDROM/refereed/597/index.html.

Neubert, S. (1994). *Modellkonstruktion in MIKE – Methoden und Werkzeuge*. Number 60 in DISKI. infix, St. Augustin. PhD Thesis.

Newell, A. (1982). The knowledge level. *Artificial Intelligence: An International Journal*, 18(1):87–127.

Nichols, D. M. & Twidale, M. B. (1999). Computer supported cooperative work and libraries. *Vine (Special Issue on Virtual Communities and Information Services)*, 109:10–15.

Nonaka, I. & Takeuchi, H. (1995). *The Knowledge-Creating Company*. University Press, Oxford.

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

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

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

Noy, N. F. & Musen, M. A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 450–455.

Noy, N., Fergerson, R., & Musen, M. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In (Dieng & Corby, 2002), pages 17–32.

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.

Noy, N. (2002). The OntoWeb evaluation experiment for ontology editors: Using Protégé-2000 to represent the travel domain. In (Sure & Angele, 2002), pages 103–107. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Odgen, C. K. & Richards, I. A. (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. (1998). Using AI in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, 13(3):34–39.

Ontoprise (2002a). How to write F–Logic programs – a tutorial for the language F–Logic. Tutorial version 1.9 that covers Ontobroker version 3.5.

Ontoprise (2002b). OntoEdit Tutorial. available through Ontoprise GmbH.

Ontoprise (2002c). OXML 2 reference manual. available at http://www.ontoprise.de/download/oxml2.0.pdf.

OntoText (2003a). BOR website. http://www.ontotext.com/bor.

OntoText (2003b). Ontology Middleware Module (OMM) demo website. http://omm.ontotext.com.

OntoText (2003c). Ontology Middleware Module (OMM) website. http://www.ontotext.com/omm.

OntoWeb (2001). OntoWeb Annex 1 – "Description of work". European Commision (EU), Brussels.

On-To-Knowledge (1999). On-To-Knowledge Annex 1 – "Description of work". European Commision (EU), Brussels.

Papakonstantinou, Y., Garcia-Molina, H., & Widom, J. (1995). Object exchange across heterogeneous information sources. In *Proceedings of the IEEE International Conference on Data Engineering, Taipei, Taiwan, March 1995*, pages 251–260.

Pinto, H. S., Peralta, D. N., & Mamede, N. J. (2002). Using Protégé-2000 in reuse processes. In (Sure & Angele, 2002), pages 15–25. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Pirlein, T. (1995). *Wiederverwendung von Commonsense Ontologien im Knowledge Engineering: Methoden und Werkzeuge*, volume 85 of *DisKI*. Infix, Sankt Augustin.

Plaza, E. & Benjamins, V. R. (Eds.) (1997). *Proceedings of the 10th European Workshop on Knowledge Acquisistion, Modeling, and Management (EKAW'97)*, volume 1319 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer.

Pocsai, Z. (2000). *Ontologiebasiertes Wissensmanagement für die Produktentwicklung*, volume 3/2000 of *Forschungsberichte aus dem Institut für Rechneranwendung in Planung und Konstruktion der Universität Karlsruhe*. Shaker Verlag.

Polanyi, M. (1958). *The Tacit Dimension*. Doubleday & Co., Garden City, NY.

Polanyi, M. (1974). *Personal Knowledge*. University of Chicago Press, Chicago.

Preece, A. (2000). *Evaluating Verification and Validation Methods in Knowledge Engineering*, pages 91–104. In (Roy, 2000).

Probst, G., Romhardt, K., & Raub, S. (1998). *Wissen managen*. Frankfurter Allgemeine Zeitung, Frankfurt am Main, Wiesbaden, Gabler, 2. aufl. edition.

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

Puppe, F. (Ed.) (1999). *XPS-99: Knowledge Based Systems – Survey and Future Directions, 5th Biannual German Conference on Knowledge Based Systems*, volume 1570 of *Lecture Notes in Artificial Intelligence (LNAI)*, Würzburg, Germany. Springer.

Quinn, J. (1992). *Intelligent Enterprise. A knowledge and service based paradigm for industry.* Free Press, New York.

Raggett, D., Le Hors, A., & Jacobs, I. (1998). HTML 4.0 Specification. W3C Recommendation 24 April 1998. available at http://www.w3.org/TR/REC-html40.

Raś, Z. W. & Skowron, A. (Eds.) (1999). *Foundations of Intelligent Systems – Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems (ISMIS'99)*, number 1609 in Lectute Notes in Artificial Intelligence (LNAI). Springer-Verlag.

Reimer, U., Abecker, A., , Staab, S., & Stumme, G. (Eds.) (2003). *Proceedings of the 2nd National Conference "Professionelles Wissensmanagement – Erfahrungen und Visionen (WM2003)"*, volume P–28 of *GI-Edition Lecture Notes in Informatics (LNI)*, Luzern, Switzerland. Gesellschaft fuer Informatik (GI).

Reimer, U. & Mahling, D. (Eds.) (2000). *Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM 2000)*, volume 34 of *CEUR Workshop Proceedings*, Basel, Switzerland. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-34/.

Rogers, J. (2002). OntoWeb travelling domain experiment: Results for OpenKnoME. In (Sure & Angele, 2002), pages 99–102. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Rossi, G., Garrido, A., & Schwabe, D. (2000). Navigating between objects. lessons from an object-oriented framework perspective. *ACM Computing Surveys*, 32(30).

Roy, R. (Ed.) (2000). *Industrial Knowledge Management: A Micro–level Approach.* Springer-Verlag, London, Berlin, Heidelberg.

Russ, T., Valente, A., MacGregor, R., & Swartout, W. (1999). Practical experiences in trading off ontology usability and reusability. In (KAW, 1999). available at http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html.

Schneider, U. (1996a). *Management in der wissensbasierten Unternehmung*, pages 13–48. In (Schneider, 1996b).

Schneider, U. (Ed.) (1996b). *Wissensmanagement.* Frankfurter Allgemeneine Zeitung, Frankfurt am Main.

Schnurr, H.-P., Staab, S., Studer, R., Stumme, G., & Sure, Y. (Eds.) (2001). *Proceedings of the 1st National Conference "Professionelles Wissensmanagement – Erfahrungen und Visionen (WM2001)"*, Berichte aus der Informatik, Aachen. Shaker Verlag.

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

Schnyder, A. B. (1989). Unternehmungskultur: Die Entwicklung eines Unternehmungskultur–Modells unter Berücksichtigung ethnologischer Erkenntnisse und dessen Anwendung auf die Innovations–Thematik. In *Europäische Hochschulschriften*, Reihe 5. Volks– und Betriebswirtschaft; 987, pages 227–243. Peter Lang Verlag.

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.

Seely-Brown, J. & Duguid, P. (1991). Organisational learning and communities of practice. *Organisational Science*, 2(1).

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

Smith, H. & Poulter, K. (1999). Share the ontology in XML-based trading architectures. *Communications of the ACM*, 42(3):110–111.

Smith, M. K., McGuinness, D., Volz, R., & Welty, C. (2002). Web Ontology Language (OWL) Guide Version 1.0. W3C Working Draft 04 November 2002, available at http://www.w3.org/TR/owl-guide/.

Smolle, P. & Sure, Y. (2002). FRED: Ontology-based agents for enabling e-coaching support in a large company. In *Second International Workshop on Ontologies in Agent Systems (OAS 2002), held at the 1st International Conference on Autonomous Agents & Multiagent Systems*, Bologna, Italy.

Sowa, J. F. (2000). *Knowledge Representation, Logical, Philosophical and Computational Foundations.* Brooks Cole Publishing Co., Pacific Grove, CA.

Spyns, P., Meersman, R., & Jarrar, M. (2002a). Data modelling versus ontology engineering. *SIGMOD Record – Web Edition*, 31(4). Special Section on Semantic Web and Data Management; R. Meersman and A. Sheth (eds.); Available at http://www.acm.org/sigmod/record/.

Spyns, P., Oberle, D., Volz, R., Zheng, J., Jarrar, M., Sure, Y., Studer, R., & Meersman, R. (2002b). OntoWeb – a semantic web community portal. In (Karagiannis & Reimer, 2002), pages 189–200.

Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R., & Sure, Y. (2000). Semantic community web portals. In *Proceedings of the 9th International World Wide Web Conference (WWW9)*, volume 33, pages 473–491, Amsterdam, The Netherlands. Elsevier.

Staab, S., Braun, C., Bruder, I., Duesterhoeft, A., Heuer, A., Klettke, M., Neumann, G., Prager, B., Pretzel, J., Schnurr, H.-P., Studer, R., Uszkoreit, H., & Wrenger, B. (1999). A system for facilitating and enhancing web search. In *Proceedings of International Working Conference on Artificial and Natural Neural Networks: Engineering Applications of Bio-Inspired Artificial Neural Networks (IWANN'99)*, volume 1607 of *LNCS*, pages 706–714, Berlin. Springer Verlag.

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):26–34.

Staab, S. & Schnurr, H.-P. (2000). Smart task support through proactive access to organizational memory. *Journal of Knowledge-based Systems*. September 2000.

Staab, S. & Schnurr, H.-P. (2002). *Knowledge and Business Processes: Approaching an Integration*, pages 75–88. In (Dieng-Kuntz & Matta, 2002).

Staab, S., Studer, R., & Sure, Y. (2003). *Knowledge Processes and Knowledge Meta Processes in Ontology-based Knowledge Management*, pages 47–68. In (Holsapple, 2003b).

Staab, S. & Studer, R. (Eds.) (2003). *Handbook on Ontologies in Information Systems*. International Handbooks on Information Systems. Springer. To appear 2003.

Stader, J. & Macintosh, A. (1999). Capability modelling and knowledge management. In *Applications and Innovations in Expert Systems VII, Proceedings of ES 99 the*

19th International Conference of the BCS Specialist Group on Knowledge-Based Systems and Applied Artificial Intelligence, pages 33–50, Cambridge. Springer-Verlag.

Steimann, F. (2000). On the representation of roles in object-oriented and conceptual modelling. *Data & Knowledge Engineering*, 35(1):83–106.

Stewart, T. A. (1997). *Intellectual Capital – The New Wealth of Organizations*. Doubleday/Currency, a division of Bantam Doubleday Dell Publishing Group, Inc.

Stojanovic, L., Maedche, A., Motik, B., & Stojanovic, N. (2002a). User-driven ontology evolution management. In (Gómez-Pérez & Benjamins, 2002), pages 285–300.

Stojanovic, L., Stojanovic, N., & Handschuh, S. (2002b). Evolution of the metadata in the ontology-based knowledge management systems. In (Minor & Staab, 2002), pages 65–77.

Stojanovic, L., Stojanovic, N., & Volz, R. (2002c). Migrating data-intensive web sites into the semantic web. In *Proceedings of the ACM Symposium on Applied Computing (SAC-02)*, Madrid, Spain.

Stojanovic, N., Stojanovic, L., & Gonzales, J. (2002d). More efficient searching in a knowledge portal – an approach based on the analysis of users' queries. In (Karagiannis & Reimer, 2002), pages 513–524.

Stuckenschmidt, H., Stubkjaer, E., & Schlieder, C. (Eds.) (2003). *The Ontology and Modeling of Real Estate Transactions in European Juristictions*. International Land Management Series. Ashgate. To appear 2003.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering principles and methods. *Data and Knowledge Engineering*, 25(1–2):161–197.

Studer, R., Sure, Y., Volz, R., Jijuan, Z., & Meersman, R. (2001a). Creation of a browsable prototype of the portal. OntoWeb deliverable 6.2, Institute AIFB, University of Karlsruhe & StarLAB, Vrije Universiteit Brussels.

Studer, R., Sure, Y., Volz, R., Jijuan, Z., & Meersman, R. (2001b). Seed ontology. OntoWeb deliverable 6.1, Institute AIFB, University of Karlsruhe & StarLAB, Vrije Universiteit Brussels.

Studer, R., Sure, Y., & Volz, R. (2002). Managing focused access to distributed knowledge. *Journal of Universal Computer Science (J.UCS)*, 8(6):662–672.

Stumme, G., Hotho, A., & Berendt, B. (Eds.) (2001). *Proceedings of the First Workshop on Semantic Web Mining*. Held in conjunction with the (De

Raedt & Flach, 2001) and (De Raedt & Siebes, 2001); Online available at http://semwebmine2001.aifb.uni-karlsruhe.de/.

Stumme, G. & Maedche, A. (2001). FCA-MFerge: Bottom-up merging of ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 225–234, Seattle USA.

Sure, Y., Akkermans, H., Broekstra, J., Davies, J., Ding, Y., Duke, A., Engels, R., Fensel, D., Horrocks, I., Iosif, V., Kampman, A., Kiryakov, A., Klein, M., Lau, T., Ognyanov, D., Reimer, U., Simov, K., Studer, R., van der Meer, J., & van Harmelen, F. (2003a). *On-To-Knowledge: Semantic Web–Enabled Knowledge Management*, chapter 13, pages 278–301. In (Zhong et al., 2003). To appear 2003.

Sure, Y., Angele, J., & Corcho, O. (2003b). Second International Workshop on Evaluation of Ontology based Tools (EON 2003). Submitted 2003.

Sure, Y. & Angele, J. (Eds.) (2002). *Proceedings of the First International Workshop on Evaluation of Ontology based Tools (EON 2002)*, volume 62 of *CEUR Workshop Proceedings*, Siguenza, Spain. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

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 held in conjunction with Confederated International Conferences: On the Move to Meaningful Internet Systems (CoopIS, DOA, and ODBASE 2002), cf. (Meersman et al., 2002)*, pages 69–78.

Sure, Y., Maedche, A., & Staab, S. (2000). Leveraging corporate skill knowledge – from ProPer to OntoProPer. In (Reimer & Mahling, 2000), pages 22:1–22:9. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-34/.

Sure, Y. & Schnurr, H.-P. (Eds.) (2003). *Proceedings of the 1st National "Workshop Ontologie-basiertes Wissensmanagement (WOW2003)"*. April 2003, Luzern, Switzerland; held in conjunction with (Reimer et al., 2003); To appear 2003.

Sure, Y., Staab, S., & Angele, J. (2002b). OntoEdit: Guiding ontology development by methodology and inferencing. In (Meersman et al., 2002), pages 1205–1222.

Sure, Y., Staab, S., & Studer, R. (2002c). Methodology for development and employment of ontology based knowledge management applications. *SIGMOD Record – Web Edition*, 31(4). Special Section on Semantic Web and Data Management; R. Meersman and A. Sheth (eds.); Available at http://www.acm.org/sigmod/record/.

Sure, Y., Staab, S., & Studer, R. (2003c). *On-To-Knowledge Methodology.* In (Staab & Studer, 2003). To appear 2003.

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

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

Sure, Y. & Studer, R. (2002a). *On-To-Knowledge Methodology*, chapter 3, pages 33–46. In (Davies et al., 2002b).

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

Sure, Y. & Studer, R. (2002c). On-To-Knowledge Methodology — final version. On-To-Knowledge deliverable 18, Institute AIFB, University of Karlsruhe.

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

Sure, Y. (2002b). On-To-Knowledge – ontology based knowledge management tools and their application. *Künstliche Intelligenz (German Journal of Artificial Intelligence)*, 2002(1). Special Issue on Knowledge Management.

Sure, Y. (2002c). Travelling domain experiment: Engineering with OntoEdit. In (Sure & Angele, 2002), pages 83–92. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

Sure, Y. (2003). *A Tool-supported Methodology for Ontology-based Knowledge Management.* In (Stuckenschmidt et al., 2003). To appear 2003.

Swartout, B., Patil, R., Knight, K., & Russ, T. (1996). Toward distributed use of large-scale ontologies. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*, Banff, Canada.

Swartout, B., Ramesh, P., Knight, K., & Russ, T. (1997). Toward distributed use of largescale ontologies. In *Symposium on Ontological Engineering of AAAI*, Stanford, CA.

Tennison, J. & Shadbolt, N. (1998). APECKS: A tool to support living ontologies. In *Proceedings of the 11th Knowledge Acquisition Workshop (KAW'98)*, Banff, Canada.

Tiwana, A. (2000). *The Knowledge Management Toolkit.* Prentice Hall PTR., Upper Saddle River, NJ, USA.

TOVE (1995). TOVE: Manual of the Toronto Virtual Enterprise. Technical report, Department of Industrial Engineering, University of Toronto. available at http://www.ie.utoronto.ca/EIL/tove/ontoTOC.html.

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

Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The enterprise ontology. *Knowledge Engineering Review*, 13(1):31–89.

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 Gelder, A., Ross, K. A., & Schlipf, J. S. (1991). The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650.

van Gelder, A. (1993). The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221.

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

Volz, R., Oberle, D., & Studer, R. (2002). Towards views in the semantic web. In *Proceedings of the 2nd International Workshop on Databases, Documents, and Information Fusion (DBFUSION 02)*, Karlsruhe, Germany.

Volz, R., Oberle, D., & Studer, R. (2003). Views for light-weight web ontologies. In *Proceedings of the 2003 ACM Symposium of Applied Computing (SAC), Melbourne, Florida.* To appear.

W3C (2001). World Wide Web Consortium (W3C) Semantic Web Activity Statement, available at http://www.w3.org/2001/sw/Activity/.

Welty, C. A. & Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering*, 39(1):51–74.

Wiederhold, G. & Genesereth, M. (1997). The conceptual basis for mediation services. *IEEE Expert / Intelligent Systems*, 12(5):38–47.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49.

Wiederhold, G. (1993). Intelligent integration of information. In *SIGMOD-93*, pages 434–437.

XML/EDI-Group (2003). XML/EDI, the E-Business framework. available at http://www.xmledi-group.org/.

Y. Jin, S. Decker, G. W. (2001). Ontowebber: Model-driven ontology-based web site management. In *Proceedings of the 1st International Semantic Web Working Symposium (SWWS'01)*, Stanford University, Stanford, CA. available at http://www.semanticweb.org/SWWS/program/full/.

Zhong, N., Liu, J., & Yao, Y. (Eds.) (2003). *Web Intelligence.* Springer-Verlag". To appear 2003.

Please note: The cited URLs were provided by the author in all conscience. They were last checked in January 2003. However, even though "cool URIs don't change" (*cf.* (Berners-Lee, 1998)), URIs in the dynamic surrounding of the WWW (typically referred to as URLs (Berners-Lee, 1993)) and the content they represent are subject to change. In future they might differ from the cited sources in this work.

Index

agent, 4
AIFB Semantic Portal, 185
annotation, 189
APECKS, 221
Application phase, 50, 156, 164
ARANEUS, 224
AutoWeb, 224
axiom, 22, 206

BOR, 141

case study, 141
  BT, 143, 158
  Communities of Knowledge Sharing, 158
  EnerSearch, 143, 168
  Skills Management, 145
  Swiss Life, 142, 144
  Virtual Organization, 168
Chief Information Officer, 13
Chief Knowledge Officer, 13
Chimera, 221
CIO, 13
CKM, 13
CKO, 13
class, 22
Client-Server Framework, 93
CMF, 190
combination, 12
CommonKADS, 34, 219
competency question, 46, 149
computer-supported cooperative work, 75
concept, 22, 69
  concept hierarchy, 69
  concept instantiation, 70
  direct subconcept, 69
conceptual structures, 68
content management, 188
CORPORUM, 135, 144
CREAM, 58
Cyc, 220

DAML, 66
DAML+OIL, 66, 70
DARPA–DAML, 223
Description Logic, 67, 80, 138, 209
DOGMA, 59, 183, 185, 188–190, 220, 222
DOGMA Server, 220, 222
DOGMAModeler, 220, 222
domain expert, 76
Domain Lexicon, 147
Domain Lexicon Plugin, 90

Edu Portal, 194
EnerSEARCHer, 143, 168, 173
Enterprise Ontology, 220
EON 2002, 182, 196
Evaluation phase, 46, 156, 162, 169
Evolution phase, 51, 156, 164
experiment, 196
externalization, 12, 15

F-Logic, 67, 70, 79
FaCT, 67
Feasibility Study, 34, 145, 158
First-Order Logic, 23

formal guidelines, 39
Frame, 138, 209
Frame Ontology, 22

General Axioms Editor, 98
Graphical Rule Editor, 98

Horn Logic, 79, 141
human issues, 31

IIPS, 224
Inferencing Plugin, 113
information technology, 13
instance, 22
internalization, 12, 15
interoperability, 196
ISWC2002, 182

KA2, 224
KA2 Community Web Portal, 185
KACTUS, 220
KAON, 117, 222
KAON Portal, 189
KAON Syndicator, 189
Kickoff phase, 37, 83, 149, 160, 199
knowledge, 3, 11
  explicit knowledge, 12
  formal knowledge, 55
  implicit knowledge, 12, 15
  informal knowledge, 55
  knowledge transfer, 12
  tacit knowledge, 11
Knowledge Process
  Knowledge Access, 53
knowledge item, 17, 54
knowledge level, 23
knowledge management, 4, 11, 12
  application, 31
  culture, 13
  organization, 13
  people, 15
  technology, 15
Knowledge Manager, 13
Knowledge Meta Process, 16, 31, 32
  Application phase, 50, 156, 164
  Evaluation phase, 46, 156, 162, 169
  Evolution phase, 51, 156, 164
  Feasibility Study, 34, 145, 158
  Kickoff phase, 37, 83, 149, 160, 199
  Refinement phase, 42, 152, 160, 199
knowledge model, 79
Knowledge Process, 16, 53
  Knowledge Creation, 53
  Knowledge Import, 53
  Knowledge Retrieval, 53
  Knowledge Access, 59
  Knowledge Capture, 53, 57
  Knowledge Creation, 54
  Knowledge Import, 56
  Knowledge Retrieval, 59
  Knowledge Use, 54, 61
knowledge sharing, 4

logic, 23
Loom, 210

Management Console, 72
meta data, 5, 18, 186
Methodology, 16, 31, 123
Methodology plugins, 74
METHONTOLOGY, 220
Mind2Onto, 42, 87, 152, 201

Natural Language Processing, 209

ODBASE2002, 182
OEE, 66
OEE requirements, 67
  collaboration, 67, 74
  extensibility, 67, 71
  inferencing, 67, 74
  methodology, 67, 74
  ontology languages, 67, 70
OIL, 66, 138

OilEd, 210, 222
OImodeller, 222
On-To-Knowledge, 122
  BOR, 141
  BT case study, 143, 158
  Communities of Knowledge Sharing, 158
  CORPORUM, 135, 144
  EnerSearch case study, 143, 168
  EnerSEARCHer, 143, 168, 173
  OIL, 138
  OntoEdit, 129, 143
  OntoExtract, 135, 142, 144
  Ontology Middleware Module, 130, 143, 144
  OntoShare, 127, 143, 159
  OntoView, 131
  OntoWrapper, 135, 142, 144
  OTK Case Studies, 122, 141
  OTK Tool Suite, 124
  QuizRDF, 125, 142, 143, 168, 173
  Sesame, 115, 133, 143, 144
  Skills Management, 145
  Spectacle, 127, 142, 143, 168, 173
  Swiss Life case study, 142, 144
  Virtual Organization, 168
OntoAnalyzer, 38, 104
Ontobroker, 67, 77, 80, 106, 184, 208
OntoClean, 39
OntoClean Methodology, 49, 107, 153
OntoClean Plugin, 39, 107
OntoEdit, 68, 129, 143, 199
  knowledge model, 79
OntoEdit plugins, 74
  Client-Server Framework, 93
  Domain Lexicon, 147
  Domain Lexicon Plugin, 90
  General Axioms Editor, 98
  Graphical Rule Editor, 98
  Inferencing Plugin, 113
  Methodology plugins, 74
  Mind2Onto, 42, 87, 152, 201
  OntoAnalyzer, 38, 104
  OntoClean Plugin, 39, 107
  OntoFiller, 91, 147
  OntoGenerator, 101
  OntoKick, 83, 150
  OntoMap, 98
  Query Tool, 113
  Rule Debugger, 114
  Sesame Plugin, 115
  Visualizer, 112
OntoExtract, 135, 142, 144
OntoFiller, 91, 147
OntoGenerator, 101
OntoKick, 83, 150
Ontolingua, 222
Ontology, 21
ontology, 4, 21
  application ontology, 26
  common sense ontology, 27
  domain ontology, 26
  ontology with lexicon, 69
  task ontology, 26
  top level ontology, 26
ontology development, 65
ontology editor, 66
ontology engineer, 76
ontology engineering, 65
ontology engineering environment, 66
Ontology Middleware Module, 130, 143, 144
ontology requirements specification document, 37, 83
OntoMap, 98
OntoMat, 71
OntoMat–*Annotizer*, 58, 190, 249
OntoMat–SOEP, 117
Ontosaurus, 222
OntoShare, 127, 143, 159
OntoView, 131
OntoWeb, 181

OntoWeb Semantic Portal, 59, 185
OntoWebber, 224
OntoWrapper, 135, 142, 144
OpenKnoME, 210
Option Manager, 72
organizational memory, 15, 16, 144
ORSD, 37, 83
OWL, 66, 141

P2P, 194
plugin, 71
  connector plugin, 73
  core plugin, 73
  GUI plugin, 73
  parser plugin, 73
plugin framework, 71
pragmatic guidelines, 38
Predicate Logic, 79
property, 22
Protégé, 38, 189, 210, 222
publishing workflow, 191

Query Tool, 113
QuizRDF, 125, 142, 143, 168, 173

RDF(S), 70
Refinement phase, 42, 152, 160, 199
relation, 22, 69
  relation hierarchy, 70
  relation instantiation, 71
Roadmap Portal, 194
roles, 39
RQL, 134, 143
Rule Debugger, 114

S-CREAM, 190
SEAL, 184
SEmantic portAL, 184
Semantic Web, 5
Semantic Web Mining, 61
SemTalk, 210
service-consumer, 73
service-provider, 73
Sesame, 115, 133, 143, 144
Sesame Plugin, 115
socialization, 12
software engineering, 32
Spectacle, 127, 142, 143, 168, 173
Strudel, 224
syndication, 189
synthetic ontology, 102

TERMINAE, 210
Tiramisu, 224
TOVE, 221

URI, 68

validation, 49
verification, 49
Visualizer, 112

WebODE, 210, 223
WebOnto, 223

XML, 186

ZOPE, 190