D7.1.2 SEKT Methodology: Initial Framework and Evaluation of Guidelines

York Sure, Christoph Tempich and Denny Vrandečić
(Institute AIFB, University of Karlsruhe (TH))
Sofia Pinto
(Instituto Superior Tecnico,
Universidade Tecnica de Lisboa, Portugal)
Elena Paslaru Bontas
(Free University of Berlin)
Mark Hefke
(FZI Research Center for Information Technologies at the University of Karlsruhe)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D7.1.2 (WP 7)
This deliverable provides the first set of guidelines for ontology engineering and application in SEKT: DILIGENT – DIstributed , Loosely-controlled and evolvInG Engineering of oNTologies.
Keyword list: Ontology Engineering Methodology — Rhetorical Structure Theory (RST) — Holistic Approach — Cost Estimation for Ontology Engineering

Document Id.	SEKT/2004/D7.1.2/v1
Project	SEKT EU-IST-2003-506826
Date	January 12, 2006
Distribution	public

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE, UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP, UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006 Madrid, Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe, Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Siemens Business Services GmbH & Co. OHG
Otto-Hahn-Ring 6
81739 Munich, Germany
Tel: +49 89 636 40 225, Fax: +49 89 636 40 233
Contact person: Dirk Ramhorst
E-mail: dirk.ramhorst@siemens.com

Empolis GmbH
Europaallee 10
67657 Kaiserslautern, Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe, Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck
Institute of Computer Science
Technikerstraße 13
6020 Innsbruck, Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH
Tal
6464 Springen, Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Sirma Group Corp., Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784, Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona, Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Changes

Version	Date	Author	Changes
0.1	13.06.05	Christoph Tempich	initial creation
0.8	30.11.05	Christoph Tempich	inclusion of cost model and elaborated detailed description
0.9	6.12.05	Christoph Tempich	Release for internal review
0.95	11.1.06	Christoph Tempich	Incorporation of reviewers comments

Executive Summary

This deliverable provides a methodology to determine an the knowledge management maturity level of an organization. This depends on the level of experience of the organization w.r.t. knowledge management from an organizational, human and technology perspective. For the technological aspects of KM the deliverable provides an elaborated set of guidelines for ontology engineering and application in SEKT: DILIGENT – DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. The deliverable consists of the following building blocks: (i) a survey of relevant existing approaches to aforementioned topics, (ii) the DILIGENT process model and argumentation framework, (iii) a cost model to estimate the ontology building efforts[1], (iv) a process model to learn ontologies and (v) guidelines for the evaluation of the DILIGENT process model.

[1]Results reported in this chapter were partly financed by a Knowledge Web T-Rex exchange

Contents

1 Introduction **1**
- 1.1 Readers Guide . . . 1
- 1.2 Motivation . . . 2
- 1.3 The SEKT Big Picture . . . 4
- 1.4 The SEKT Methodology - a Fine-grained View . . . 4
- 1.5 Organization of the Deliverable . . . 6

2 Survey **9**
- 2.1 The Methodology Focus . . . 9
  - 2.1.1 Definition of Methodology for Ontologies . . . 11
  - 2.1.2 Set of Procedures . . . 12
  - 2.1.3 Documenting the Results . . . 13
  - 2.1.4 Step-by-step "Cookbook" . . . 13
  - 2.1.5 Set of Criteria for Evaluation . . . 13
- 2.2 Arguments . . . 13
  - 2.2.1 Visualizing Argumentation . . . 13
  - 2.2.2 Argumentation Theory . . . 14
- 2.3 Ontology Learning Processes . . . 17
- 2.4 Existing Tools . . . 17
  - 2.4.1 Tools for Visualization of Arguments . . . 17
  - 2.4.2 Ontology Engineering Tools . . . 19
  - 2.4.3 Conclusions . . . 21
- 2.5 Past and Current Research . . . 21
  - 2.5.1 Visualization of Argumentation . . . 21
  - 2.5.2 Methodologies . . . 25
  - 2.5.3 Cost Estimation in Ontology Engineering . . . 31
  - 2.5.4 Conclusions . . . 31

3 The HIKNOW maturity level analysis **35**
- 3.1 Introduction . . . 35
- 3.2 Methodological Approach . . . 38
  - 3.2.1 Procedure of Identifying an Organization's KM Maturity Level . . . 38
  - 3.2.2 Kochikar's KM Maturity Model . . . 39

3.3 A Conceptual Data Model for Representing KM Maturity Models 40
3.3.1 Description of Ontology Terms 41
3.3.2 Retaining KM Maturity Models 41
3.3.3 Identification of a Company's Maturity Level 42
3.4 Related Work 43
3.5 Conclusion and Future Work 43

4 DILIGENT Process and Argumentation Framework 45
4.1 Motivational Scenario 45
4.2 DILIGENT Overview 46
4.3 Detailed Process Description: Use Case Oriented 47
4.3.1 Local Adaptation: Detailed View 48
4.3.2 Analysis 51
4.3.3 Revision 53
4.3.4 Local Update 54
4.4 Generic Detailed Process Description 55
4.4.1 Building 56
4.4.2 Local Adaptation 60
4.4.3 Analysis 68
4.4.4 Revision 71
4.4.5 Local Update 75
4.5 Requirements for DILIGENT Tool Support 78
4.5.1 Build 79
4.5.2 Local Adaptation 81
4.5.3 Analyzing 82
4.5.4 Revision 83
4.5.5 Local update 83
4.6 Argumentation Framework for DILIGENT 85
4.6.1 Threads of Arguments 85
4.6.2 RST Example 85
4.6.3 Analysis in the Biology Domain 86
4.6.4 Evaluation of Argumentation Framework 88
4.7 An Argumentation Ontology for DILIGENT Processes 93
4.7.1 The argumentation process 94
4.7.2 Use Case 97
4.7.3 Requirements 99
4.7.4 Argumentation Ontology Description 100

5 ONTOCOM cost estimation model 105
5.1 Introduction 105
5.2 The ONTOCOM Cost Model 107
5.2.1 Cost Drivers for Ontology Building 109
5.2.2 Cost Drivers for Ontology Reuse and Maintenance 110

5.2.3 Evaluation of ONTOCOM . . . 110
5.2.4 Current Limitations . . . 112
5.3 Bridging ONTOCOM and DILIGENT . . . 112
5.3.1 Mapping the Activities in DILIGENT to the Cost Drivers in ONTOCOM . . . 113
5.3.2 Changes in the Cost Model . . . 117
5.4 A Cost Function for DILIGENT Processes . . . 124
5.4.1 The Complete Cost Function . . . 124
5.4.2 The Reduced Cost Function . . . 128
5.4.3 Applications of the Reduced Cost Function . . . 130
5.5 Data Collection and Model Calibration . . . 132
5.5.1 Technical Realization of the Data Collection . . . 133
5.5.2 Calibration Method . . . 133
5.6 Conclusions . . . 140

6 DILIGENT ontology learning process 143
6.1 Motivation . . . 143
6.2 Process . . . 143
6.2.1 Feasibility Study . . . 144
6.2.2 Requirements Specification . . . 147
6.2.3 Selection of Information Sources . . . 148
6.2.4 Selection of Ontology Learning Tools . . . 148
6.2.5 Learning Preparation . . . 149
6.2.6 Learning Execution . . . 150
6.2.7 Ontology Evaluation . . . 151
6.2.8 Ontology Integration . . . 152
6.2.9 Future Work . . . 152

7 Evaluation of the DILIGENT methodology 153
7.1 Selected Methods for Evaluation . . . 154
7.1.1 Goal Free . . . 155
7.1.2 Professional Review . . . 156
7.1.3 Case Study . . . 156
7.1.4 Other Approaches . . . 157
7.2 Goal Free Evaluation of DILIGENT . . . 158
7.3 Professional review . . . 159
7.3.1 DILIGENT process evaluation . . . 159
7.3.2 Argumentation framework evaluation . . . 160
7.4 Case Study Evaluation: First Application of DILIGENT . . . 161

8 Conclusions 165

List of Figures

1.1 The SEKT Big Picture 5

3.1 Methodology for the holistic introduction of KM (HIKNOW) 36
3.2 Excerpt of KM Maturity Model Ontology 40
3.3 Hierarchization of models 42

4.1 Roles and functions in distributed ontology engineering 46
4.2 Process stages (1-5), actions (1-17) and structures 48
4.3 Process stages (1-5), activities (1-32) and structures 57
4.4 Local Adaptation: Activity Diagram 62
4.5 Analysis: Activity Diagram 69
4.6 Revision: Activity Diagram 73
4.7 Local Update: Activity Diagram 76
4.8 The major concepts of the argumentation ontology and their relations . . 101

5.1 ONTOCOM data collection: introductory questions 134
5.2 ONTOCOM data collection: cost drivers 134
5.3 Data export from phpESP 135

6.1 Stages, Roles and Activities in ontology engineering with ontology learning 145

List of Tables

2.1 Summary of ontology engineering methodologies taken from [GPFLC03] 26

3.1 Level-Organizational Capability Mapping, Source: [Koc00] 39

4.1 Arguments used and outcome 91
4.2 Types of conflict according to Shaw and Gaines, 1989 97
4.3 List of possible inconsistencies 98

5.1 Cost Driver LEXP (Language Experience) 109
5.2 Mapping DILIGENT to ONTOCOM: Centralized Building 114
5.3 Mapping DILIGENT to ONTOCOM: Local Adaptation 116
5.4 Mapping DILIGENT to ONTOCOM: Centralized Analysis and Revision 118
5.5 Mapping DILIGENT to ONTOCOM: Local Update 119
5.6 The Domain Complexity Cost Driver DCLPX 120
5.7 The Conceptualization Complexity Cost Driver CCPLX 121
5.8 The implementation complexity cost driver ICPLX 121
5.9 Ratings for Documentation Costs 122
5.10 The Ontology Evaluation Cost Driver OE 122
5.11 The Ontology Integration Cost Driver OI 123
5.12 The Tool Support Cost Driver TOOL 124
5.13 Simplified cost model factors 135
5.14 Delphi result 136
5.15 Data collection 136
5.16 Adjusted collected data 137
5.17 Correlation matrix for our example 138
5.18 Results of the linear regression 138
5.19 Parameter estimation from experts and based on the data 138
5.20 Effort estimation based on expert estimation and historical data 140
5.21 Results of the linear regression - alternative 140

6.1 ORSD for the travel domain 146
6.2 Evaluation of tool feasibility 149

7.1 Summary of ontology engineering methodologies adapted from [GPFLC03] 158

Chapter 1

Introduction

The SEKT work package 7 "SEKT methodology" consist of four deliverables, which will describe a methodology to apply the SEKT technologies in a knowledge management setting. This deliverable is the extension of deliverable 7.1.1 "Initial Guidelines" and describes the process related issues. Deliverable 7.2.1 reports on the application of this process in the case studies. Before we describe the SEKT methodology we illustrate in this chapter the overall organization of this deliverable. We first comment on the history of this deliverable and then motivate our work and situate it in the overall SEKT project. This is followed by an overview of the SEKT methodology. Finally we outline the organization of the deliverable.

1.1 Readers Guide

The combined deliverables in work package 7 will result in a a guideline to apply and use the technologies developed in the SEKT-project in a knowledge management setting. The document evolves in the course of the project. In the first year we outlined initial guidelines. In the second year we elaborated on the initial guidelines and collected first experiences in the application of those guidelines. The last year we will work on the validated methodology with extended case study experiences.

In order to emphasize the evolving nature of this work, we have decided to include the changes, updates and elaborations for the methodology in the existing deliverable. The new sections are highlighted in black while the old parts appear in dark gray. The reader familiar with the old version can thus distinguish between new and old work while the reader new to the field has all information available in one document. In particular we update the state of the art part and introduce references to recent works in our area. Furthermore, we integrate the HIKNOW methodology, offering a holistic approach to knowledge management and integrating the ontology engineering methodology in a general framework of knowledge management. We elaborate on the process model itself and

include a generic description and introduce an argumentation ontology. A major extension is the alignment of the methodology with the ONTOCOM cost estimation model. The total costs of ownership of DILIGENT developed ontologies can be estimated with this model. Initial results for an ontology engineering process model by ontology learning are presented in the penultimate chapter.

Even though the final methodology will contain academic methodological work as well as case study experiences these two aspects are still separated at this stage of the project. The interested reader will find our first experiences with the methodology in the related deliverable *7.2.1 First best practices* [VSTE05].

1.2 Motivation

The SEKT project 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 work.

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.* [Qui92]). **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."*
> [Dru93]

This so-called *"post-industrial revolution"* (*cf.* [Jac96]) focused the view on knowledge as *"intellectual capital"* (*cf.* [Ste97]) 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.* [EM97]).

Knowledge management (KM) emerged as a 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."*
> [Kay03]

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."*
> [DP98]

Successful KM strategies consist of building blocks for organization, people, technology and corporate culture (*cf.* [Alb93, Sch96a]. In such a context, knowledge sharing is not only a matter of communication between humans, but also between them and machines. Interacting agents (human and software agents) need to share their knowledge, thus requiring a common language. Generalizing the quotation above we might say that *"Agents can't share knowledge if they don't speak a common language"*.

Ontologies were exploited in Computer Science to enhance knowledge sharing and reuse (*cf.* [Gru93, Fen01]). 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.* [SS03]).

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. Similarly 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.*,

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

(ii) **extracting information** is left to 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 [DFv02]).

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."
[BLHL01]

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

1.3 The SEKT Big Picture

This report is part of the work performed in workpackage (WP) 7 on "Methodology". As shown in Figure 1.1 this work is closely related with "Usability and Benchmarking"; they both are intermediating between the research and development WPs and the case study WPs in SEKT.

The main goal of this activity is to provide a methodology for the application of Semantic Knowledge Technologies into enterprises in a goal-driven and process-oriented way. Hence, we need an integrated approach to knowledge management balancing the organisation and management aspects on the one hand, and the information technology and systems aspects on the other hand.

Central to Semantic Knowledge Technologies is the application of ontologies to knowledge management problems. Core aspects for the methodology therefore include the efficient and effective creation and maintenance of ontologies in settings such as provided by the case studies.

1.4 The SEKT Methodology - a Fine-grained View

SEKT seeks to integrate advantages of the different technologies mentioned in the previous section (*cf.* fig. 1.1) for the overall goal to provide better technological support for knowledge management.

The work described in WP 7 is concerned with the methodological aspects of integrating the different technologies and thereby overcome weaknesses of currently available methodologies.

1. Classical development of knowledge-based systems and of corresponding ontologies is mostly *centralized* like the targeted knowledge-based system itself. In contrast, we here consider the general tendency to support *distributed information processing with ontologies*, *e.g.* the Semantic Web, agents, web services or ontology-based peer-to-peer. Stakeholders in an ontology development process will

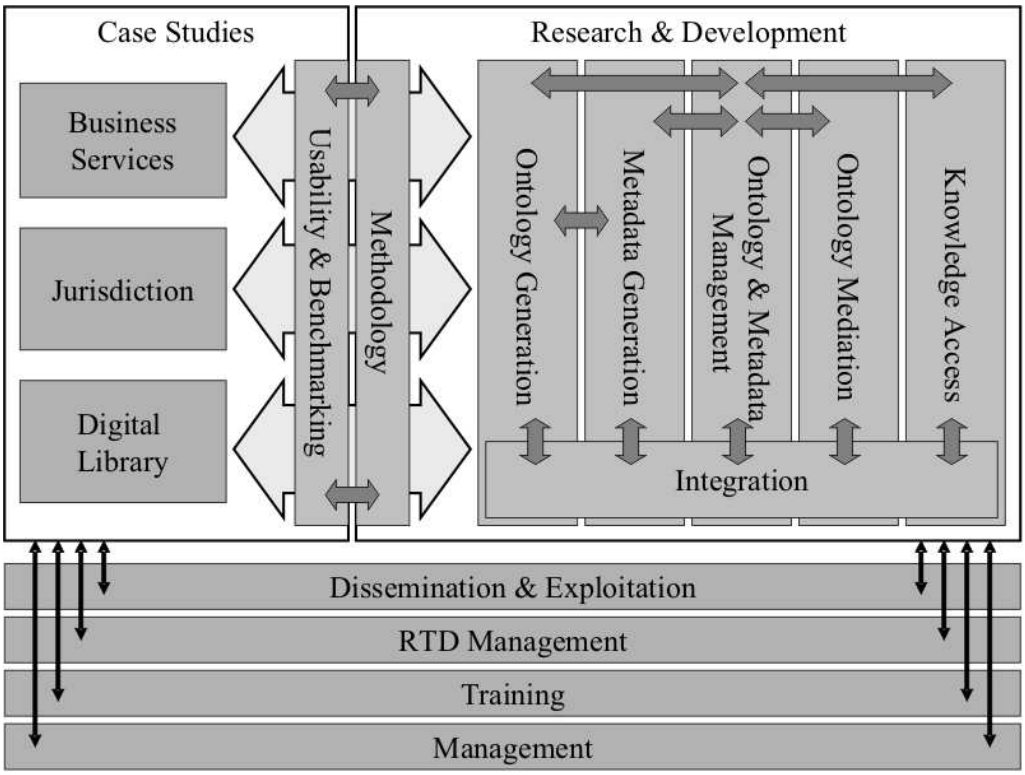

Figure 1.1: The SEKT Big Picture

hardly ever gather in one place. Yet they have an interest to contribute fruitfully toward the ongoing development of their ontologies.

2. Existing methodologies support knowledge engineering (KE) by using check lists that guide the engineering process. The check lists have been shaped by the needs of *knowledge engineers* to cope comprehensively with nearly arbitrarily complex processes. In contrast, in the distributed cases we consider, the participation of a knowledge engineer is often restricted to a, possibly complex, core ontology. Beyond the core, these cases involve extensive participation and, comparatively simple, concept formation by *domain experts*.

3. KE has mostly focused on an *up- and running system* with some moderate effort for maintenance. In contrast, ontologies for distributed information processing must permanently *evolve* in order to reflect the widely diverging needs of their users.

4. KE methodologies remain rather *coarse* and the gap between their description and concrete actions to be taken is filled by the KE. In contrast, for Semantic Web ontologies and comparable use cases, we ask the question whether we could provide the domain experts with *fine-grained* guidance in order to improve their effectiveness and efficiency in ontology engineering.

To account for some of the differences between classical knowledge engineering and ontology engineering methodologies derived from there, we have started to develop a methodology for DIstributed (cf. item 1 above), Loosely-controlled (cf. item 2) and evolv-InG (cf. item 3) Engineering of oNTologies, the validity of which has been partially checked and is still being checked against experiences in two case studies (cf. [PSST04].

The methodology will combine aspects from different areas to integrate the different technologies in a concise way. We argue that this requires a major abstraction step which leads us to a methodology applicable in distributed environment, which can only be loosely controlled and evolves constantly. At this abstraction level we could recognize similarities between our objectives and objectives being worked on in the field of argumentation visualization. Without going into detail we here list the two areas which we will subsequently analyse.

- Methodologies for ontology engineering
- Computer supported argumentation

In the reminding report we will refer in each chapter to these areas. The overview chapter will explain the different parts and relate them to the overall objective of the deliverable.

1.5 Organization of the Deliverable

This deliverable is organized as follows. We first present our results of a review of related work in the area in chapter 2. We have in particular analyzed the current state of the art of ontology engineering tools and tools for argumentation visualization. On the research side we have identified the existing methodologies for ontology engineering. Furthermore we have summarized the relevant research areas for argumentation visualization and conflict resolution with a focus on ontology engineering. From this review we could derive several open issues with respect to (w.r.t.) ontology engineering methodologies.

Before we elaborate on the DILIGENT methodology we present the HIKNOW maturity level analysis in chapter 3. Introducing knowledge management starts seldom in the open countryside but most of the times some effort has already been undertaken. As SEKT technologies provide solutions for challenging knowledge management problems, we first have to determine the current state of affairs in a company in order to introduce the technologies appropriately. The HIKNOW maturity level analysis provides the meas to determine the maturity of a company w.r.t. knowledge management.

In Chapter 4 we motivate and present the DILIGENT process and argumentation framework. We show a first evaluation of the process in in-situ experiments at the Institute AIFB. Finally, we elaborate on our initial studies for the argumentation framework, i.e. a

thorough ex-post analysis of an evolving taxonomy in the biology domain from which we derived our argumentation framework based on the Rhetorical Structure Theory (RST).

In Chapter 5 we describe the ONTOCOM cost estimation model. Current ontology engineering methodologies do not account for or allow the user of the methodology to estimate the effort spend to build an ontology. We have aligned the DILIGENT methodology with the ONTOCOM cost estimation model in order to overcome this weakness in our methodology. Besides the alignment we elaborate on the process of refining ONTOCOM to provide accurate effort estimates.

In Chapter 6 we present initial procedural guidelines to ontology engineering by means of ontology learning. The process model is part of the overall DILIGENT process and elaborates on the specific part of engineering by learning.

Before concluding we illustrate in Chapter 7 how we initially applied DILIGENT in the SEKT case studies and provide ideas for further application and adaptation.

Chapter 2

Survey

This chapter provides a survey of related work. It is organized as follows. First, we describe the focus of our methodology, provide a definition for methodology and explain how this relates to ontologies in Section 2.1. We concentrate on two major areas which have impact on our methodology, viz. arguments in Section 2.2 and ontology learning in Section 2.3. Then we provide a more detailed view on existing tools (Section 2.4) and past and current research on ontology engineering methodologies (Section 2.5).

2.1 The Methodology Focus

It has been a widespread conviction in knowledge engineering that methodologies for building knowledge-based systems help knowledge engineering projects to successfully reach their goals in time (cf. [SAA+99] for one of the most widely deployed methodologies). With the arrival of ontologies in knowledge-based systems the same kind of methodological achievement for structuring the ontology-engineering process has been pursued by approaches to OE like [GPFLC03, SSSS01, UK95] and their application has been proposed in such areas as the Semantic Web, too. At this point, however, we have found some mismatches between these proposals (including our own) and the requirements we meet in the Semantic Web.

In the next sections we will describe different areas which we currently regard as related to our work. Before we will explain, why these areas are important. Therefore we recall the objective of the SEKT methodology: to provide support for the ontology engineer in the task of creating, maintaining and instantiating an ontology with the help of the SEKT technologies. Hence, it is immediately clear that existing work in the area of ontology engineering methodologies is relevant in our context. The SEKT technologies are mainly concerned with automation of the ontology creation task, *i.e.* "ontology learning". The processes which have been defined to support the ontology learning task must therefore be integrated into a common methodology. From a wider perspective an ontology

learning method can be seen as an agent which proposes changes to a given ontology or creates a new one. The ontology engineer must then decide which changes are integrated in the ontology. This is very similar to a setting in which many people collaboratively build an ontology for a given domain. In a collaborative environment participants propose changes, exchange arguments and finally agree. The capturing of arguments is analyzed as part of the area of "Visualizing Argumentations". We can use the experiences made in that field to integrate ontology learning methods with the work of a human ontology engineer. Additionally we can make the ontology engineering decisions more transparent. To make the methodology more general we not only consider synchronous collaboration but also asynchronous and distributed collaboration. Our methodology should not require that the ontology is designed in one location. We explicitly want to support distributed engineering of ontologies. With this abstraction we can consequently handle the engineering of ontologies in distributed settings given in peer-to-peer systems, but also the automated engineering of ontologies using different Web services or users working with a centralised system such as Livelink where each users has its own view (extension) to an ontology.

For the research related to our work we looked with varying intensity for related work. In the case of "ontology engineering methodologies" we did an extensive review of research in that area, as it is at the core of our research. Related work on "ontology learning" can be found in the deliverables of work package 3, therefore we refer the interested reader to those deliverables. We did also search for related work on "visualizing argumentations" and "argumentation" in general. While we cite for the ontology engineering case also work which is not supported any more or has been integrated into other methodologies in the case of argumentation we concentrate on the main contributions, as our objective is to use this technology for our means.

One of the main features of automated methods is that they can be applied repeatedly with very low additional costs. By the incorporation of automated methods into the ontology engineering task, we can apply those methods continuously to available data. This data will change and so will the ontology. Evolution and change of the ontology must thus be explicitly dealt with in our methodological framework.

Compared to existing ontology design methods, introducing collaboration, distribution and evolution significantly reduces the amount of precision and control available in the process of producing the ontology. Analogously to people designing an ontology, different automated methods will propose different extensions to an ontology. An initially shared ontology may develop in different departments of a company or in various areas of the web in various directions just as organisms have found many ways to adapt to our environment through evolutions. However, from an ontological point of view it is desirable to share the conceptualization. The methodology must organize the different processes in a way that the participants can find the highest common denominator.

2.1.1 Definition of Methodology for Ontologies

> **methodology**[1] – An organised, documented set of procedures and guidelines for one or more phases of the software life cycle, such as analysis or design. Many methodologies include a diagramming notation for documenting the results of the procedure; a step-by-step "cookbook" approach to carry out the procedure; and an objective (ideally quantified) set of criteria for determining whether the results of the procedure are of acceptable quality.

The first experiences in building ontologies have lead to the introduction of different methodologies to support the process of building. The first methodologies like [GF95b, UG96] described particular experiences in building ontologies for specific applications. [2]
METHONTOLOGY is an established and generalized ontology engineering methodology. We will follow their terminology, as there the terminology introduced by the IEEE for software development methodologies is related to ontology engineering. Before the terms *methodology, method, technique, process, activity* were used without a precise definition. We describe the terminology in the following.

The IEEE defines a **methodology** as *"a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought be performed*[IEE90]

A **method** is a set of *"orderly process or procedure used in the engineering of a product or performing a service*[IEE90]. A **technique** is *"a technical and managerial procedure used to achieve a given objective*[IEE90].

A **process** is a function *"function that must be performed in the software life cycle. A process is composed of activities* [IEE96]. An **activity** is *"a constituent task of of a process* [IEE96]. A **task** *"is a well defined work assignment for one or more project members. Related tasks are usually grouped to form activities*[IEE96].

The difference between activity and task is not clear cut. Depending on the projected size of the ontology, the number of participants etc. a task which might be performed in one step for a small ontology might be an activity in other cases. In order to stay clear we have defined only a small number of activities for each process stage and depicted them in Activity Diagrams. The tasks are then outlined in the detailed description for each activity.

Following this citation a methodology should contain the four items:

- Procedures
- Documentation standards

[1] see `http://computing-dictionary.thefreedictionary.com/Methodology`
[2] See related work section were we provide a more detailed description of those works.

- Guidelines (cookbook)
- Evaluation criteria and analysis techniques

The definition of a methodology is now applied to the ontology engineering setting. In the following we separate the different parts which constitute a methodology and present the objectives in each step for the ontology engineering problem. We will subsequently analyse for the related work which part the according methodology supports, and are able to draw a picture in which the missing points emerge.

2.1.2 Set of Procedures

In the ontology building, procedures for the following three activities must be defined:

- Ontology management activities
- Ontology development oriented activities
- Ontology support activities

Ontology management activities: Procedures for ontology management activities must include definitions for the scheduling of the ontology engineering task. Further it is necessary to define control mechanism and quality assurance steps.

Ontology development oriented activities: When it comes to the development of the ontology it is important that procedures are defined for enrolling environment and feasibility studies. After the a decision to build an ontology the ontology engineer needs procedures to specify, conceptualize, formalize and implement the ontology. Finally the users and engineers need guidance for the maintenance/population, use and evolution of the ontology.

Ontology support activities: To aid the development of an ontology, a number of important supporting activities should be undertaken. Supporting activities include knowledge acquisition, evaluation, integration, merging and alignment and configuration management. These activities are performed in all steps of the development and management process. Knowledge acquisition can happen in a centralized as well as a decentralized way. Ontology learning is a way to support the manual knowledge acquisition with machine learning techniques.

2.1.3 Documenting the Results

It is important to document the results after each activity. In a later stage of the development process this helps to trace why certain modelling decisions have been undertaken. The documentation of the results can be facilitated with an appropriate tool support. Depending on the methodology the documentation level can be quite different. One methodology might require to document only the results of the ontology engineering process while others give the decision process itself quite some importance.

2.1.4 Step-by-step "Cookbook"

Each of the analysed methodologies provides some sort of step by step "cookbook". However, they differ in the requirement on the ontology engineer. It is desirable that even a "rookie" ontology engineer could refine and extend an ontology after studying the current status. However, in most cases the methodologies are not fine grained enough to enable untrained persons to engineer an ontology from scratch. For each of the activities a step-by-step "cookbook" should define the input data and output data and the exact procedures how to transform input into the desired output.

2.1.5 Set of Criteria for Evaluation

In the ontology engineering setting evaluation measures should provide means to measure the quality of the created ontology. This is particular difficult for ontologies, since modelling decisions are in most cases subjective. A general survey of evaluation measures for ontologies can be found in [GP04]. Additionally we want to refer to the evaluation measures which can be derived from statistical data (*cf.* [TV03]) and measures which are derived from philosophical principles. One of the existing approaches to ontology evaluation is OntoClean [GW02] which *e.g.* has been implemented as part of OntoEdit [SAS03]. This approach is based on philosophical principles, so far only few examples of its application to practical use cases are known. Further research on ontology evaluation is currently being carried out as part of the EU IST thematic network Knowledge Web[3].

2.2 Arguments

2.2.1 Visualizing Argumentation

We can deploy research in the area of argumentation visualization [CSSS01, KSE03] to solve certain requirements on the envisioned SEKT methodology. Argumentation visual-

[3] see `http://knowledgeweb.semanticweb.org/`

ization helps its users to discuss their problems in a clearly defined way. The methodologies proposed in this field base their recommendations on different models of agreement processes. Following the methodologies various tools have been implemented. The main advantage of computer supported argumentation is the achieved traceability of decisions. In particular the traceability problem is well researched in the software engineering community (*cf.* [PB88]).

> **Traceability** – "A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation" (ANSI/IEEE Standard 830-1984) [IEE84].

The original response to argumentation visualization was the application of the IBIS methodology [KR70]. IBIS allows users to capture different design deliberations. Appropriate tools (see Section 2.4) can help to retrieve them in a sophisticated manner later on. However, the IBIS technique has received criticism due to its resilience to change and being too abstract.

In our context different actors collaborate to design an ontology and find commonalities. They will exchange arguments in favor or against certain modelling decision. We believe that by selecting the right model for argumentation we can on the one hand enhance the traceability of modelling decisions and on the other hand guide the engineering in a fine grained way towards a final shared ontology. However, simply applying for example the IBIS methodology will probably not be sufficient [PB88]. According to his findings IBIS should be enhanced with domain specific knowledge. More recently [GF97] has found that further enhancement of IBIS can be achieve by introducing an acceptance and rejection mechanism.

2.2.2 Argumentation Theory

Besides an intuitive and correct way to present the exchanged arguments to the user, the underlying argumentation theory is also a main concern. There are a number of argumentation theories ranging from informal explanations of argumentation threads to very formal specifications. To exemplify the notion of argument we introduce here the oldest model of natural argumentation. This provides an idea of the general concepts uses in this area. In the subsequent chapters we will present current theories and tools which have been developed bearing in mind these theories.

The Toulmin model [TRJ84], a natural argumentation theory that tries to explain how real people (not philosophers) argue, has its main components: **Data** (facts, data and information, the reason for the claim), **Claim** (the position on the issue, the conclusion being advocated) and **Warrant** (logical connection between the data and the claim, the

reasoning process used to arrive at the claim,[4]). Other components are: **Backing** (material supporting the warrant), **Reservation** (exceptions to the claim) and **Qualifiers** (relative strength).

The following examples are taken from the ChangingMinds website[5].

Claim A claim is a statement one persons asks another one to accept. This includes information they should accept as true or actions they should accept and enact.

For example:

> You should use a hearing aid.

Many people start with a claim, but then find that it is challenged. To convince another person one must prove the claim. This is where grounds become important.

Grounds The grounds (or data) is the basis of real persuasion and is made up of data and hard facts. It is the truth on which the claim is based. The actual truth of the data may be less than 100%, as all data is based on perception and, hence, has some element of assumption about it.

It is critical to the argument that the grounds are not challenged, because if they are, they may become a claim, which must be proven with even deeper information and further argument.

For example:

> Over 70% of all people over 65 years have a hearing difficulty.

Data Data is usually a very powerful element of persuasion, although it does affect people differently. Those who are dogmatic, logical or rational will more likely to be persuaded by data. Those who argue emotionally and who are highly invested in their own position will challenge it or otherwise try to ignore it. It is often a useful test to give something factual to the other person that disproves their argument and watch how they handle it. Some will accept it without question. Some will dismiss it out of hand. Others will dig deeper, requiring more explanation. This is where the warrant comes into its own.

Warrant A warrant links data to a claim, legitimizing the claim by showing the data to be relevant. The warrant may be explicit or unspoken and implicit. It answers the question 'Why does that data mean your claim is true?'

For example:

> A hearing aid helps most people to hear better.

[4] Authoritative, motivational, and substantive (which includes cause-effect, effect-cause, generalization based on example, classification, etc.).

[5] see `http://changingminds.org/disciplines/argument/toulmin.htm`

The warrant may be simple and it may also be a longer argument with additional sub-elements, including those described below.

Backing The backing (or support) to an argument gives additional support to the warrant by answering different questions.

For example:

Hearing aids are available locally.

Qualifier The qualifier (or modal qualifier) indicates the strength of the leap from the data to the warrant and may limit how universally the claim applies. They include words such as 'most', 'usually', 'always', 'sometimes'. Arguments may thus range from strong assertions to fuzzy statements.

For example:

Hearing aids help most people.

Reservation Another variant is the reservation, which may give the possibility of the claim being incorrect.

For example:

Unless there is evidence to the contrary, hearing aids do no harm to ears.

Qualifiers and reservations are much used by advertisers who are constrained not to lie. Thus they slip 'usually', 'virtually', 'unless' and so on into their claims.

Rebuttal Despite the careful construction of the argument, there may still be counter-arguments that can be used. These may be rebutted either through a continued dialogue, or by pre-empting the counter-argument by giving the rebuttal during the initial presentation of the argument.

For example:

There is a support desk that deals with technical problems.

Any rebuttal is an argument in itself, and thus may include a claim, warrant, backing and so on. It also, of course can have a rebuttal. Thus if you are presenting an argument, you can seek both possible rebuttals and also rebuttals to the rebuttals.

2.3 Ontology Learning Processes

Ontology learning aims at the integration of a multitude of disciplines in order to facilitate the construction of ontologies, in particular ontology engineering and machine learning. Because the fully automatic acquisition of knowledge from machines remains in the distant future, the overall process is considered to be semi-automatic, i.e. with human intervention. It relies on a coordinated interaction between human modeler and learning algorithm for the construction of ontologies.

In [Mae02] a generic ontology learning architecture is presented. The process model there builds on the principal idea of data mining as a process (*e.g.* [CKC+99]) with the phases of business and data understanding, data preparation, modeling, evaluation and deployment.

Ontology learning is being recognized as an important topic and numerous people are moving to focus on the topic[6]. However, no holistic approach to ontology learning similar to or an extension of the previously mentioned one is known so far. We will continue to survey this rapidly emerging field to take into account the newest research results.

2.4 Existing Tools

2.4.1 Tools for Visualization of Arguments

Within the SEKT project argumentation visualization is not a primary research focus. We rather want to use the mature ideas from that field to enhance the ontology engineering process. Therefore we omit a complete analysis of available tools and just refer the interested reader to [GF94]. There over 100 commercial tools and research products were reviewed. However, we here summarize the main findings from there analysis. The issue raised provides a fruitful input for our own research.

A number of problems in the field of traceability are identified. Surprisingly, the inability to locate and access the sources of requirements is the most commonly cited problem across all the practitioners in there investigations. This problem was also reported to cause many others:

- An out of date RS (Requirements specification), as an RS evolves poorly when those originally responsible are not involved in its evolution, or where it is impossible to regain the original context.
- Slow realization (and deterioration as a result) of change, as the most time-consuming and erroneous part is often the identification of those to involve and

[6] see e.g. the workshops on (i) Mining for and from the Semantic Web (MSW) at `http://km.aifb.uni-karlsruhe.de/ws/msw2004` and (ii) Ontology Learning and Population (OLP) at `http://olp.dfki.de/ecai04/cfp.htm` for very first approaches to ontology learning

inform.

- Unproductive conflict resolution, decision making, and negotiation, as most tools supporting these activities do not help to identify or locate the essential participants.

- Poor collaboration, as the invisibility of changing work structures and responsibilities makes it difficult to: transfer information amongst parties; integrate work; and assign work to those with relevant knowledge and experience.

- Difficulty in dealing with the consequences when individuals leave a project and with the integration of new individuals.

- Poor reuse of requirements, as reuse is mainly successful when those initially responsible for their production are either directly involved or readily accessible.

2.4.1.1 Selected Tools for Argument Visualization

As the number of available tools in this area is huge we pick out only one commercial tool based on the most famous system to capture deliberations. From the research tools we choose the ones most cited in the research community.

QuestMap QuestMap is an award-winning product for mediating meetings through Visual Information Mapping. QuestMap originates from the pioneering hypertext system building by Jeff Conklin in the mid-1980s, whose team at MCC developed gIBIS[CB88] for capturing software design rationale, and then went onto build QuestMap.

Compendium Compendium [SSS+01] builds on the gIBIS methodology. It is a semantic hypertext tool to capture arguments and visualize them. It offers a conceptual framework of argumentation, it promotes the use of a meeting facilitator and there exist a number of tools to present the exchanged arguments to different audiences. Compendium tools include *Question based templates* to facilitate the flow of the arguments. Hence, the discussion can be lead by "pre-formulated" questions which structure the discussion. The process of the discussion is visualized by different maps, interlinking and connecting the exchanged arguments. In Compendium any kind of idea can be expressed since its notation is very flexible.

ClaiMaker ClaiMaker [KSE03, LUM+02] focuses on scientific debate where scientists can express the positions and contributions in a publication through a combination of free text and structuring constructs.

Tellis The Tellis tool [BG04, GR02] is a system for structured argumentation on any topic where users progress from information sources to arguments that intermix free text and structured connectors.

2.4.2 Ontology Engineering Tools

An early overview of tools that support ontology engineering can be found in [DSW+00]. However, there have been joint efforts of members of the thematic network OntoWeb[7], who provided an extensive state-of-the-art overview on ontology related tools, including Ontology Engineering Environments (OEE, *cf.* [GPAFL+02]). A sign for the growing interest in Ontologies and tools that support ontology engineering is the (recently updated) published comparison of ontology editors on XML.com (*cf.* [Den02, Den04]). An evaluation of ontology engineering environments has been performed as part of the EON 2002 workshop (*cf.* [SA02]). With respect to our work in SEKT, especially the following tools are noteworthy.

APECKS [TS98] 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 [MFRW00] 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.* [Hal01]) as graphical notion and its cross-bonding ORM-ML to ensure easy exchange (*cf.* [DJM02]). It supports the database-inspired DOGMA ontology engineering approach and is coupled with the DOGMA Server as a backend.

KAON OImodeller [MMV02, BEH+02] belongs to the KAON tool suite. 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. Recently the OWL-DL and SWRL reasoning engine KAON2`http://kaon2.semanticweb.org/` has been added to the KAON landscape of tools. The extension of the KAON tool suit is part of the SEKT project.

OilEd [BHGS01]is a graphical ontology editor that initially was dedicated to modelling of DAML+OIL (now OWL) 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 [Hor98] to classify ontologies and check consistency via translation from OWL to the SHIQ description logic. However, the tool is not extensible *e.g.* by plugins, nor does is provide sophisticated support for collaboration aspects.

[7] see `http://www.ontoweb.org/`

The **Ontolingua** [FFR96] 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 OWL, nor with the state-of-the-art technology.

Ontosaurus [SPKR96] 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 [NFM00] 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 OWL. Recently also support for axioms was added through the "PAL tab" (Protégé axiom language, *cf.* [HNM02]).

WebODE [ACFLGP01] 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.* [GP96, FLGPSS99]). 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 OWL.

WebOnto [Dom98] 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.

OntoEdit [SEA+02a, SAS03] supports explicitly the OTK Methodology [Sur03]. The open plug-in framework enables the integration of a number of extension to the basic ontology management services OntoEdit provides. In particular OntoEdit offers advanced support for collaboration and a integration of the inferencing capabilities. Noteworthy is the plug-in which implements the recommendations of the OntoClean methodology [GW02]. A re-implementation of OntoEdit based on IBM's eclipse framework has recently been made available under the new name OntoStudio[8].

HCOME [KVA04] HCOME is probably the most recent development in the field

[8] `http://www.ontoprise.de`

of ontology management tools in the research community. The tool supports a scenario which is very similar to the objectives of the SEKT methodology. The tool supports the usual ontology management functions such as versioning and editing. However, it is not clear if they support inferencing. Besides those rather traditional functions, HCOME supports the discussion of ontological decisions with support of the IBIS methodology [KR70]. The ontology engineers may work distributively on different ontology, and are supported in collaboratively engineering a shared ontology. Their methodology does not include support for automated methods to ontology engineering, neither exists an elaborated process to reach agreement towards a shared ontology.

2.4.3 Conclusions

There exists a plethora of ontology engineering tools. Major critique points from our point of view are the following ones.

- The tools typically target manual ontology creation without integration of automatic approaches to ontology learning.
- Only very few tools provide support for distributed engineering of ontologies.
- None of the tools supports structured argumentation during the creation of ontologies.
- Apart from KAON no tool supports properly the evolution of ontologies.

2.5 Past and Current Research

2.5.1 Visualization of Argumentation

As we could already see while analysing the available tools for argumentation visualization, the number of them is huge. Similarly the number of argumentation models and related research is very diverse. We attempt nevertheless to structure the area and provide references to the research we will deploy for our methodology.

Therefore we distinguish several research areas which contribute to the field in a whole. An important aspect of argumentation is the mode – **synchronous, asynchronous** – in which it is performed. Different **models** have been developed to conceptualize the way argumentation is done. Furthermore the **conceptualization of arguments** themselves is subject to investigation. In an argumentation **conflicts** can arise, thus models of conflict exist and proposals how to resolve them systematically. We recall that our objective is to deploy the findings from the selected areas to enhance ontology engineering. Hence, we finally analyse the work done in the **intersection** of argumentation and ontology engineering.

2.5.1.1 Synchronous and Asynchronous Argument Exchange

We start with separation when and how the argumentation takes place. One can distinguish synchronous and asynchronous interaction. Synchronous interaction implies that the parties discuss the *Claims* at the same time (not necessarily at the same place). Asynchronous interaction refers to discussions where *Claims* can be brought forward at various points in time. The most obvious example are discussion per email. Asynchronous interaction is typically more difficult to support than synchronous.
[SK93] analysis which kind of arguments are exchanged in a discussion depending whether it is performed asynchronous or synchronous. Therefore they distinguish arguments related to the content, meeting management and project management. Their main findings are that issues stated in one mode are continued in that mode and that the mode was chosen according to the urgency of the required decision. Alternatives to content related issues are presented five times as often as issue themselves.
The Compendium tool [SSS+01] offers support for both modes. However, they assume that discussions take place in synchronous mode with the help of the facilitator. Subsequent discussions can then be performed partly asynchronously. The support comes mainly through visualizing the synchronous discussion in various ways.

2.5.1.2 Argumentation Model

The Toulmin model of argumentation was the first one presented in the literature. Today IBIS is the most often used model to describe argumentation. When it comes to the analysis of texts the Rhetorical Structure Theory is most often used. In the SEKT project a tool will developed by BT to automatically identify the underlying rhetorical structures of a text.

- **Information based information systems (IBIS):** IBIS (pronounced "eye-bis") stands for Issue-Based Information System, and was developed by Horst Rittel and colleagues during the early 1970's [KR70]. IBIS was developed to provide a simple yet formal structure for the discussion and exploration of "wicked" problems. Problems that are wicked, as opposed to tame, do not yield to the traditional "scientific" approach to problem solving, which is to gather data, analyse the data, formulate a solution and implement the solution for the problem. With a wicked problem your understanding of the problem is evolving as you work on a solution for it. One sure sign of a wicked problem is that there is no clear agreement about what the "real problem" is. Wicked problems cannot be solved in the traditional sense, because one runs out of resources (time, money, energy, people, etc.) before perfect solutions for them can be implemented.

  gIBIS [CB88] focuses on capturing collaborative deliberations about design in the form of graphs containing text at their nodes. It is the first graphical interface for the IBIS method.

In IBIS the following terms are used to classify different arguments.

Question / Issue States a question, raises an Issue

Idea proposes a possible resolution for the question

Argument states an opinion or judgement that either supports or objects to one or more ideas

response to Indicates a response to a question

supports Supports an argument

objects to Objects to an argument

Specializes Defines a question with more detail

Challenges Challenges an argument, an idea or a question

Justification Justifies an argument, an idea or a question

Expands-on Adds new information to an idea

Decision nodes Indicate that a decision was reached on a certain issue

- **Rhetorical Structure Theory (RST):** The aim of Rhetorical Structure Theory (RST) [MT87] is to offer an explanation of the coherence of texts. It is assumed that for every part of a coherent text there is some function. RST focuses on showing an evident role for every part of a text. A text is usually divided into structures, building blocks. These blocks are of 2 levels: nuclearity and relations. The most frequent structure is two spans of text (virtually adjacent). These are usually related such that one of them has a specific role relative to the other: the span making the claim is the nucleus (N) and the span with the evidence is the satellite (S). Thirty relations between 2 spans of text have already been identified and loosely defined. [Mar97] presents an algorithm, which is able to extract the relations from natural language text with high precision.

  For the sake of completeness we here list some of the most important relations found in RST: Elaboration, Evaluation, Justification, Contrast, Alternative, Example, Counter Example, Background knowledge, Motivation, Summary, Solutionhood, Restatement, Purpose Condition, Preparation, Circumstance, Result, Enablement, List.

In the DILIGENT methodology we will rely on RST, a brief example for our notation while using RST can be found in Section 4.6.2.

2.5.1.3 Formal Arguments

The formalization of arguments is a big topic in the AI community. Even though OWL provides us with the necessary formalism to be able to state arguments in a formal way we

do not believe that ontological decisions can be discussed in a completely formal way. At least not if the ontology is to be used by humans. [GK97] for example proposes a formal model of argumentation, using IBIS as argumentation model. With the formal model it is possible to derive a user's preferred solution for the issues based on the provided arguments. Another interesting application is the selection of arguments based on the user needs. In [Hun04] a formal model is presented how formal argumentation trees can be pruned to best correspond to the users wishes.

2.5.1.4 Conflict Mediation

[Eas91] has summarized comprehensively the field of conflict mediation. He gives an introduction to economic and behavioural models to conceptualize and resolve conflicts in discussions. For our future work it is particularly interesting which kinds of conflicts can arise in the area of knowledge acquisition.

[SG89] compares the entity-attribute models of different experts, and identify four types of comparisons between conceptual systems:

- **Consensus:** experts use the same terminology to describe the same concepts
- **Correspondence:** experts use different terminology to describe the same concepts
- **Conflict:** experts use the same terminology to describe different concepts
- **Contrast:** experts use different terminology to describe different concepts

Each of these situations can be useful in capturing different perspectives, and in particular, the availability of alternative terminologies makes a knowledge-base more accessible.

Given these different types which can lead to conflict [Eas91] propose a methodology to resolve the conflicts. The first step is to establish correspondences between different conceptualizations. Afterwards conflicting issues must be identified. They can be discussed using a system like gIBIS. The conflicting issues should be explained externalizing the assumptions behind the decisions and justifying them. Thus goals and motivations become clear to all participants. For conflicting issues resolution criteria should be defined. In a next phase the participants must generate resolution options to resolve the different conflicts. Given the evaluation criteria for the different issues, one can select the best resolution criteria for each issue.

2.5.1.5 Arguments in Ontology Engineering

The application of argumentation models to ontology engineering is still in its infancy. In [SMD02] a case study in engineering an ontology from the combination of three existing

ones is described. The compendium tool is used to guide the discussion in a synchronous meeting. The results of the case study show that structured argumentation in beneficial for ontology engineering. The traceability of the decisions was enhanced. However, the authors were more concerned with the evaluation of their tool than with the specific issues arising in an discussion about an ontology. The authors do not examine which kinds of arguments are exchanged and how the discussion could be made more efficient.

The authors of [ASvE04] propose and evaluate a three-phased knowledge mediation procedure which is especially conceived to integrate different perspectives and information needs into one consensual ontology. The knowledge mediation procedure consists of three main phases. In the generation phase users are jointly brainstorming about relevant concept and instances of the knowledge domain to outline the content of the ontology. During the explication phase each user independently works out a taxonomy by adding definitions and relations to the collected concepts. In the integration phase the knowledge mediator supports the users to integrate their proposed taxonomies into a shared conceptualization. They test the procedure with and without a moderator. With a moderator the participants exchange more elaborated arguments and try to structure their arguments better. They identify useful questions which can guide the actors in the ontological discussion. However, they do not analyse the dominant types of arguments which are used in the discussion.

KUABA The Kuaba design rational ontology [dMSF05] is a formal model of the IBIS argumentation vocabulary. It is used to capture design rationales in the Software Design domain. The Kuaba ontology is represented in OWL. It contains concepts and relations to capture *Arguments*, *Questions*, *Ideas*, *Decisions* and other entities in order to model Software design rationales. The use of the ontology shall enhance reusability and traceability and makes them machine processable. Kuaba does not define a subset of arguments particulary suitable for software design. Neither do the authors report on a case study evaluation.

2.5.2 Methodologies

An extensive state-of-the-art overview of methodologies for ontology engineering can be found in [GPFLC03] from where Table 2.1 has been taken. The book is partially the result of a joint efforts of the OntoWeb[9] members, who produced an extensive state-of-the-art overview of methodologies for ontology engineering (*cf.* [GPFLC$^+$02, FLGPE$^+$02]). There exist also deliverables on guidelines and best practices for industry (*cf.* [LAB$^+$02, LBB$^+$02]) with a focus on applications for E-Commerce, Information Retrieval, Portals and Web Communities. Jones et. al compiled a review on ontology engineering methodologies in the end of the nineties [JBCV98]. More recently [CC05] proposed a framework to compare ontology engineering methodologies and evaluated the established ones accordingly. A very practical oriented description to start building ontologies can be found

[9] see `http://www.ontoweb.org/`

Feature			Cyc	Uschold & King	Grüniger & Fox	KACTUS	METHON-TOLOGY	SENSUS	On-To-Knowledge	HCOME
Ontology management activities	Scheduling		NP	NP	NP	NP	Proposed	NP	Described	NP
	Control		NP	NP	NP	NP	Proposed	NP	Described	NP
	Quality assurance		NP	NP	NP	NP	NP	NP	Described	NP
Ontology development oriented activities	Pre development processes	Environment study	NP	NP	NP	NP	NP	NP	Proposed	NP
		Feasibility study	NP	NP	NP	NP	NP	NP	Described	NP
	Development processes	Specification	NP	Proposed	Described in detail	Proposed	Described in detail	Proposed	Described in detail	Proposed
		Conceptualization	NP	NP	Described in detail	Proposed	Described in detail	NP	Proposed	Proposed
		Formalization	NP	NP	Described in detail	Described	Described	NP	Described	Proposed
		Implementation	Proposed	Proposed	Described	Proposed	Described in detail	Described	Described	Proposed
	Post development processes	Maintenance	NP	NP	NP	NP	Proposed	NP	Proposed	Described
		Use	NP	NP	NP	NP	NP	NP	Proposed	Described
		Evolution	NP	NP	NP	NP	NP	NP	NP	Proposed
Ontology support activities	Knowledge acquisition		Proposed	Proposed	Proposed	NP	Described in detail	NP	Described	NP
	■ Distributed know. acquisition									Proposed
	■ Onto. Learning integration									
	Evaluation		NP	Proposed	Described in detail	NP	Described in detail	NP	Proposed	NP
	Integration		Proposed	Proposed	Proposed	Proposed	Proposed	NP	Proposed	NP
	Configuration management		NP	NP	NP	NP	Described	NP	Described	NP
	Documentation		Proposed	Proposed	Proposed	NP	Described in detail	NP	Described	NP
	■ Results									
	■ Decision process									Proposed
	Merging and Alignment		NP	NP	NP	NP	NP	NP	NP	Proposed

Table 2.1: Summary of ontology engineering methodologies taken from [GPFLC03]

in [NM01].

With respect to our work, especially the following approaches to ontology engineering from Table 2.1 are noteworthy. If appropriate we provide pointers to tools mentioned in the previous section, whenever tool support is available for a methodology.

CommonKADS [SAA+99] 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 [LG90] arose from experience of the development of the Cyc knowledge base (KB)[10], 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[11]. 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 its 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, *i.e.* the Cyc top-level ontology was separated into smaller modules.

DOGMA is one of the more recent ontology modeling approaches [JM02, SMJ02]. The database-inspired approach to ontology engineering 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 modeling approach is implemented in the DOGMA Server and accompanying tools such as the DOGMAModeler tool set.

The **Enterprise Ontology** [UK95] [UKMZ98] 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 the OTK Methodology and extended them. Explicit tool support is given by the Ontolingua Server, but actually these principles heavily influenced the design of most

[10] Cyc knowledge base, see `http://www.cyc.com`

[11] HPKB, see `http://reliant.teknowledge.com/HPKB/about/about.html`

of the more advanced ontology editors.

The **KACTUS** [BLC96] approach to ontology engineering 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 [GP96, FLGPSS99] 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 of the ontology development process with identification of the main activities (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.

SENSUS [SRKR97] is a top-down and middle-out approach to derive 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 [UG96] proposes a formalized method for building ontologies based on competency questions. The approach to build ontologies using competency questions, that are the questions an ontology should be able to answer, is very helpful and integrated it in OTK Methodology.

HOLSAPPLE In [HJ02] a methodology for collaborative ontology engineering is proposed. The aim of their work is to support the creation of a static ontology. A knowledge engineer defines an initial ontology which is extended and changed based on the feedback from a panel of domain experts. The feedback is collected with a questionnaire. The knowledge engineer examines the questionnaires, incorporates the new requirements and a new questionnaire is send around, until all participants agree with the outcome. Their methodology does not support synchronous collaborative ontology engineering and neither the evolution of ontologies.

HCOME In [KV03, KVA04] the authors present a very recent approach to ontology development. HCONE stands for Human Centered ONtology Environment. It supports the development of ontologies in a decentralized fashion. They introduce three different spaces in which ontologies can be stored. The first one is the *Personal Space*. In this space users can create and merge ontologies, control ontology versions, map terms and word senses to concepts and consult the top ontology. The evolving personal ontologies can be shared in the *Shared Space*. The shared space can be accessed by all participants. In the shared space users can discuss ontological discission based on the IBIS [KR70] model. After a discussion and agreement the ontology is moved to the *Agreed space*.

OTK Methodology In [Sur03] the OTK Methodology is described. This methodology is the result of the EU project OnToKnowledge. The OTK Methodology divides the ontology engineering task into five main steps. Each step has numerous sub-steps, requires a main decision to be taken at the end and results in a special outcome. The phases are "Feasibility Study", "Kickoff", "Refinement", "Evaluation" and "Application & Evolution". The sub-steps of the e.g. "Refinement" are "Refine semi-formal ontology description", "Formalize into target ontology" and "Create prototype" etc. The documents resulting from each phase are e.g. for the "Kickoff" phase an "Ontology Requirements Specification Document (ORSD)" and the "Semi-formal ontology description" etc. The documents are the basis for the major decisions that have to be taken at the end to proceed to the next phase, e.g. whether in the "Kickoff" phase one has captured sufficient requirements. The major outcomes typically serve as decision sup- port for the decisions to be taken. The phases "Refinement - Evaluation - Application - Evolution" typically need to be performed in iterative cycles. One might notice that the development of such an application is also driven by other processes, e.g. software engineering and human issues. All steps of the methodology are supported by tools available for OntoEdit. In a nutshell the OTK Methodology completely describes all steps which are necessary to build an ontology for a centralized system. However the methodology does not cover scenarios where the participants are distributed in several locations. It provides no guidance for systematically evolve an ontology and it does not incorporate automated methods for ontology creation.

CO_4 is a protocol to build consensual ontologies in a distributed setting and provides a solution for a similar setting as the one we are aiming at. It is supported by the CO_4 system [Euz95, Euz97]. The starting point for this system are a number of knowledge bases (KB) which are distributed but depend on each other hierarchically. The closer a knowledge base is to the root knowledge base the more consensual knowledge it contains. The KBs at a lower level can send requests for consensus building to higher level KBs. The higher level KBs will then distribute the request to all KBs below in the tree and collect their replies. If all KBs accept the change request it becomes part of the consensual knowledge. The KBs can also comment on the request and the proposer can reply to the comments. The protocol is concerned with the technical and formal aspects of the agreement process. It does not specify the way comments should be provided. The protocol defines only activities related to the formal model of the knowledge base, it is not concerned with the specification or knowledge acquisition related activities. The protocol was evaluated in two case studies.

IDEF5 is an ontology building methodology to support the creation of ontologies [BMM+94] for centralized settings. It is well documented. It originates and is applied by a company and is therefore not published on academic conferences. The methodology is divided into five main activities: *Organize and Define Project*, *Collect Data*, *Analyze Data*, *Develop Initial Ontology* and *Refine and Validate Ontology*. The organization and definition activity defines the managerial aspects of the ontology development project. During the collect data activity the domain analysis is performed and knowledge sources

are determined and exploited. The result of the analyze data activity is a first conceptualization of the domain. In the following activities the ontology engineers start defining so called *Proto-Concepts* which are high level concepts characterizing the domain. These are refined with relations and axioms. The Proto-Concepts are later refined until the validation results in an ontology which mets the requirements set in the beginning.

UPON the Unified Process for ONtology building, has been proposed in [NNM05]. Although the methodology has not been well tested in projects yet, and tool support is still in its infancy, it is conceptually well founded. It is based on the Unified Software Development Process and supported by UML (Unified Modeling Language). UPON defines a series of work flows which are cyclically performed in different phases. The work flows are (1) Requirements identification, e.g. by writing a story board and using competency questions, (2) Analysis, which includes the identification of existing resources and the modeling of the application scenario, (3) Design and conceptualization, (4) Implementation and finally (5) Test, in which the coverage of the application domain should be guaranteed and the competency questions are evaluated. The work flows are followed in the four phases (1) Inception, (2) Elaboration, (3) Construction and (4) Transition defined in the methodology. These phases are performed in a cyclic manner. After each cycle an applicable ontology is produced.

Hsemann et. al [eV05] present an ontology engineering methodology which is based on database schema design processes. They subdivide the task of ontology building into four subprocess: *Requirements analysis*, *Conceptual Design*, *Logical Design* and *Physical Design*. *Documentation* and *Evaluation* are support activities which are performed in parallel to each of the building activities. Each building activity has a predefined output, namely a *requirements specification* as result of the Requirements analysis, a *Conceptual schema*, a *Logical schema* and finally a *Physical schema* as a result of the Physical Design. Each of the activities is described in detail; they combine a number of approaches to *e.g.* knowledge elicitation from other methodologies with evaluation methods from database design. The methodology is supported by the OntoMedia tool. The methodology does not treat evolutionary or distributed aspects of ontology engineering.

Misc For the sake of completeness and without a detailed description we here reference some other proposals for structured ontology engineering. Among them are [PM01] advocating an approach to ontology building by reuse. One of their major findings was that current methodologies offer only limited support for axiom building even so it is a part of ontology engineering which takes a lot of time. In [GPS98] the authors outline the ONIONS approach to ontology integration. ONIONS (ONtologic Integration Of Naïve Sources) creates a common framework to generalize and integrate the definitions that are used to organize a set of terminological sources. In other words, it allows to work out coherently a domain terminological ontology (a terminological ontology is usually defined as the explicit conceptualization of a vocabulary) for each source, which then can be compared with the others and mapped to an integrated ontology library. Jones et. al. [JBCV98] list additionally the MENELAS ontology, the Mikrokosmos ontology and the PHYSSYS approach to ontology engineering. Those were early approaches to ontology

development and their findings have been included into more general methodologies such as the OTK methodology or METHONTOLOGY.

2.5.3 Cost Estimation in Ontology Engineering

Cost estimation methods have a long-standing tradition in more mature engineering disciplines such as software engineering or in industrial production [Boe81, Kem87, Ste95]. Although the importance of cost issues is well-recognized in the ontology engineering community, no cost estimation model for ontology engineering is available so far. Effort estimation models for the development of knowledge based systems (e.g. [Fel04]) or object oriented information systems (e.g. [ACC+98])depend on the availability of the conceptual model. [Men99] analysis qualitatively the costs and benefits of ontology use in applications, but does not offer any model to estimate effort. [Kor05] adjusts the cost factors defined in a cost estimation model for web applications w.r.t. the usage of ontologies. The cost factors, however, are not adapted to the requirements of ontology engineering and no evaluation is provided.

2.5.4 Conclusions

From our point of view argumentation visualization is mature from the research perspective. First attempts were made to combine findings from argumentation visualization and ontology engineering. However, as it is argued in [PB88, dMA03] argumentation is best supported when the argumentation model such as IBIS is customized with respect to the domain which is argued about. Hence, research is moving into the following directions.

- Identify the most relevant arguments in ontological discussions.
- Support synchronous as well as asynchronous discussions.

We have surveyed a number of ontology engineering methodologies and their main strength. We conclude that none of the existing methodologies covers all aspects of ontology engineering and that there still exist many open issues. Most of the methodologies address the engineering of a single ontology for a particular application and do not support maintenance activities after the first release. The methodologies proposed more recently do treat the evolution of the ontology seriously. However, while early phases of the development process are well understood and detailed activity descriptions exists (e.g. how to create competency questions), more fine-grained guidelines for later stages in the process are still missing. For example, methodological support for the creation and evolution of complex logical axioms is still an open issue. The number of best practices describing concrete development efforts is still very small. With this ontology engineering approach the quality of the development and maintenance effort depends mainly on the capabilities of the actors involved. In particular for multi site development efforts no clear guidelines

exist as here multiple views should be considered. Multiple views on the same domain, can lead to different conceptualizations, making agreement on a shared one therefore particularly difficult.

Another important aspect of ontology building, namely the construction of ontologies with the help of automated methods is not directly supported by any of the existing methodologies. Although the quality of ontology learning methods with regards to the usability of results has increased tremendously in the past years, the selection of appropriate input information or the integration of the produced ontologies with manual ones is still not well understood.

The number of existing methodologies covering different aspects of the ontology building process, suggest developing a 'method engineering' approach as has happened in software engineering. Instead of constructing new methodologies for different application scenarios, the methodology itself could include a step in which the engineers pick from a list of available methods, e.g. for requirements analysis, the ones suitable for a particular task and build up their own process model. Template process models covering standard requirements could be available.

Furthermore current methodologies are not completely integrated in a broader process model covering e.g. human, technological and process aspects of knowledge management. Although the OTK and CommonKADS methodology consider in an early phase business environment issues. These aspects are important to deploy knowledge management in a holistic manner. In this context the costs incurred by the ontology building effort become an issue. Estimating these costs is still a vague businesses and none of the existing methodologies provides guidelines for this activity.

To summarize, the following issues are still open:

1. Ontology maintenance support
2. Distributed ontology engineering
3. Fine grained guidelines for all phases
4. Representation of multiple views
5. Agreement support under conflicting interests
6. Best practices
7. Ontology engineering with the help of automated methods
8. Process definition by single process step combination
9. Integration into business process model
10. Cost estimation and pricing

This chapter has been able to survey only a few of the ontology engineering tools. Since the introduction of the plug-in concept to OEEs, the number of features available for the more established tools has increased tremendously and for many tasks one can find a tool supporting it. However, integration with different process models is still lacking. Some tools offer support for a specific process model, but none can be customized to provide guidance through an arbitrary process. As there are many tools offering different functionalities the slightly different implementations for the standardized representation languages hinders inter-operability. Besides these more procedurally oriented requirements, the technical solutions to support e.g. versioning, ontology learning or distributed engineering of ontologies, are also in need of improvement.

Chapter 3

The HIKNOW maturity level analysis

The work presented in this chapter is based on [HK05].

We describe an ontology-based software infrastructure for retaining and maintaining theoretical Knowledge Management Maturity Models (ONTOKNOM^3) by the use of a KM Maturity Model Ontology. Moreover ONTOKNOM^3 provides technical means for designing a web-based system, that supports the form-based self-evaluation of an organization w.r.t. its current maturity level, as well as for providing concrete organizational recommendations and measures in order to achieve a higher one.

3.1 Introduction

There is no doubt that Knowledge Management (KM) has to be integrated in daily business in order to handle typical knowledge processes like the acquisition, structuring, development and distribution of knowledge in a more efficient way. A look at KM literature shows that there has already been spent a lot of work on developing methods and instruments for supporting the introduction and the steady use of KM [DP98], [PRR99]. Furthermore it is clear, that KM introduction necessarily has to focus on organizational, technical and human aspects in equal measure by considering already existing organizational structures, technical infrastructure and (knowledge intensive) processes [MAAY03].

Figure 3.1 introduces the HIKNOW methodology that provides an overview of what has to be at least considered when accomplishing a successful introduction of holistic KM in an organization. The major tasks to be considered are arranged along a time line from the early project initiation over project execution and controlling until the end of the project and the transcending continuous improvement. In reality, when bringing such a KM introduction methodology to practice, one is faced with organizations that have a varying level w.r.t. already realized KM activities. At the first level we find organizations only sensitized with KM, but aware of the benefits KM could cause. Businesses on the second level others have already established first methods and solutions for dealing with

typical knowledge problems like information overflow, knowledge capturing and distribution. Finally, there are organizations that run sophisticated knowledge management systems and have well organized their KM activities. But there are also organizations that have a brilliant KM system from a technical point of view but do not have its organization in place. They realize after its deployment, that no one is using the system, because the employees were not integrated in the system planning and development process, as a result the system does not meet the real requirements of the work force. In the described cases, it is not possible to start KM again from the beginning, but rather to extensively adapt a continuous implementation methodology to the real organizational needs. Therefore it is necessary to identify by all means the current KM maturity level of an organization before starting a KM implementation project in order to avoid from the first KM project failures which are associated with unexpected costs and time to somehow rescue them.

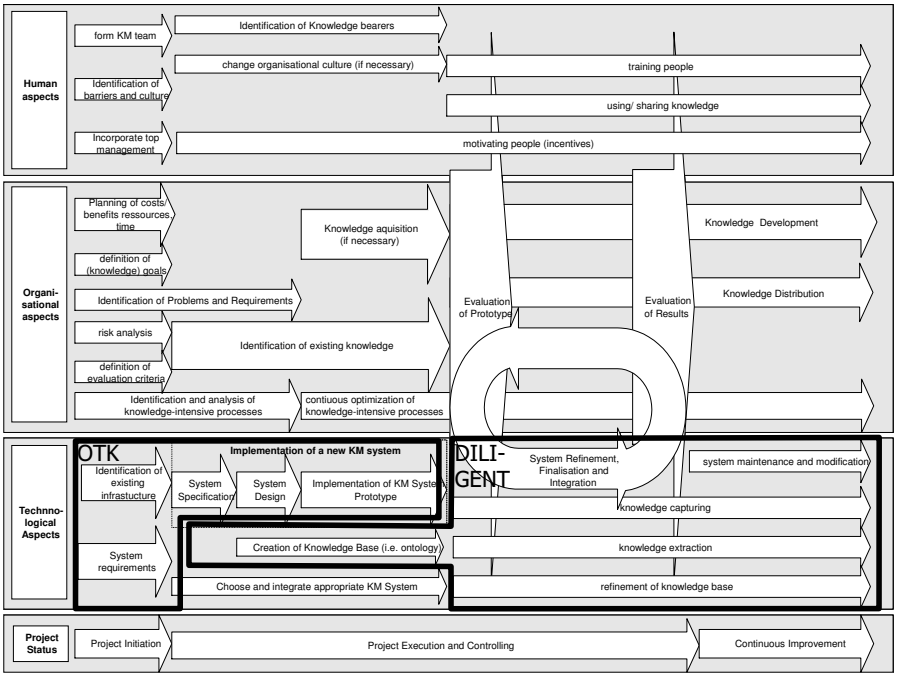

Figure 3.1: Methodology for the holistic introduction of KM (HIKNOW)

The basic idea of identifying an organization's KM maturity level derives from software engineering, where the Capability Maturity Model® for Software (SW-CMM®) [PWCC95] has been developed in order to evaluate the quality of software development processes in an organization by the use of a survey-based evaluation method. The model distinguishes between the following five maturity levels, in terms of an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes:

1. **Initial.** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and

heroics.

2. **Repeatable.** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

3. **Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

4. **Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5. **Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Except for Level 1, each maturity level is decomposed into several key process areas that indicate the areas an organization should focus on to improve its software process. In the meanwhile, there exist various theoretical KM maturity models, that are based on the original idea of SW-CMM®. An overview is given in the section 3.4. Our basic idea was not to develop yet another theoretical KM maturity model, but to design ONTOKNOM^3, a flexible and ontology-based system infrastructure for supporting the easy storage and maintenance of any theoretical KM maturity model into an ontology. Further requirements for such a system were the support for a web-based determination of an organization's current and evolving maturity level with regard to KM. This will be done by analyzing an organization's structure and technical infrastructure using question and answering, like it's done in any CMM audit. Moreover, ONTOKNOM^3 should help organizations in reaching a higher maturity level w.r.t. KM by providing concrete organizational recommendations and measures based on the underlying KM maturity model. Therefore we concentrated on the identification of an existing preferably all-embracing theoretical KM maturity model, which in addition focuses on a holistic KM introduction. After analyzing several KM maturity models we finally selected Kochikar's maturity model [Koc00] as a suitable basis for developing our model.

The rest of this chapter is organized as follows: Our methodological approach comprising the definition of a procedure of identifying an organization's KM maturity level as well as a detailed description of the selected KM maturity model from Kochikar is presented in section 3.2.2. Section 3.3.1 describes our developed KM Maturity Model Ontology, which is needed for retaining theoretical KM maturity models (levels, questions and associated measures), organization profiles, as well as an evolving organization's maturity level of an evaluating organization. After a discussion of related work in section 3.4, concluding remarks outline some future work in section 3.5.

3.2 Methodological Approach

Our methodological approach has been divided into firstly defining a typical procedure of identifying an evolving organization's maturity level to be supported by ONTOKNOM[3] and secondly analyzing Kochikar's model, in order to derivate requirements to the KM Maturity Model Ontology as well as to our software infrastructure to be developed.

3.2.1 Procedure of Identifying an Organization's KM Maturity Level

As a result of defining the typical procedure of identifying an organization's KM maturity level, we obtained the following seven major steps:

1. Firstly, the selected representative of the organization that is directly involved in the KM implementation process has to point out, if he is doing an evaluation process for the first time or if he would like to continue an ongoing evaluation.

2. If the user is doing an evaluation for the first time, he registers with the system by choosing a password and answering some general questions that are describing her company (i.e. company size, sector, etc.). After that, the system returns a company identifier for logging in.

3. If the user is continuing an ongoing evaluation process, he is asked for her company identifier and password. After that the system presents questions of the current maturity level to be answered.

4. The user selects the provided questions and gives answers to them by the use of a web-based form. If the user exits the system during an ongoing evaluation, the state of the evaluation is automatically saved and restored upon continuation.

5. After answering all questions of a maturity level, the system calculates the evaluation results in the form of a maturity level on a scale from m to n (where m represents the maturity model's start level and n represents the model's stop level). After that, the current maturity level is provided to the user and combined with recommendations about how to reach a higher one.

6. After e.g. printing out the evaluation results/recommendations, the user logs out and puts them into practice.

7. Upon completion of all displayed measures, the user logs in again and continues the evaluation process by answering the previously failed questions. As soon as all questions of a certain level are answered correctly, the level is considered as reached and the questions of the next level are presented.

3.2.2 Kochikar's KM Maturity Model

The KM maturity model presented by V.P. Kochikar [Koc00] is based on the above-mentioned SW-CMM®. The model consists of the five KM maturity levels Default, Reactive, Aware, Convinced and Sharing. Each of these maturity levels "is characterized by certain observable capabilities along each of the three major prongs People, Process and Technology. Table 3.1 depicts the five maturity levels of the KMM model.

Level		Organizational Capability
Level 1	Default	• Complete dependence on individual skills and abilities
Level 2	Reactive	• Ability to perform tasks constituting the basic business of the organization repeatable
Level 3	Aware	• Restricted ability for data-driven decision-making • Restricted ability to leverage internal expertise • Ability to manage virtual teams well
Level 4	Convinced	• Quantitative decision-making for strategic and operational applications widespread • High ability to leverage internal and external sources of expertise • Organization realizes measurable productivity benefits through knowledge sharing • Ability to sense and respond proactively to changes in technology and business environment
Level 5	Sharing	• Ability to manage organizational competence quantitatively • Strong ROI-driven decision making • Streamlined process for leveraging new ideas for business advantage • Ability to shape change in technology and business environment

Table 3.1: Level-Organizational Capability Mapping, Source: [Koc00]

Moreover the model has a set of Key Result Areas (KRAs) for each level. Each KRA is specific to one of these three "prongs" and represents at a given maturity level the organization's KM capability.

For example on Level 3 the KRAs for the prongs are, for people: ”Central Knowledge Organization”, ”Knowledge Education”, for the process: ”Content Structure Management”, and for technology: ”Knowledge Technology Infrastructure”.

3.3 A Conceptual Data Model for Representing KM Maturity Models

In order to achieve a conceptual data model for representing one or more theoretical KM maturity model(s), that are consisting of different levels, questions and linked measures as well as for retaining and managing organization profiles that have to be associated with evolving maturity levels, we modeled an ontology [SS04], which is providing the data layer.

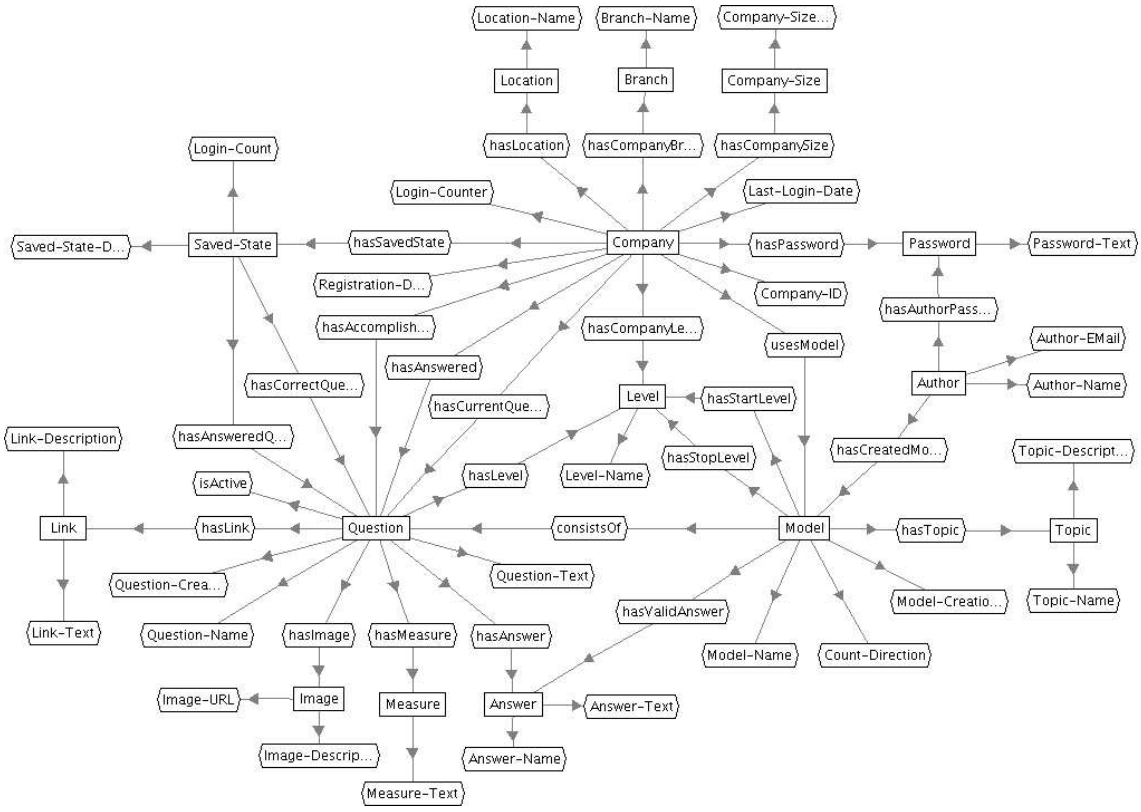

Figure 3.2: Excerpt of KM Maturity Model Ontology

3.3.1 Description of Ontology Terms

The KM Maturity Model Ontology is modeled in KAON language and contains, among others, the top concepts ”Company”, ”(Maturity-)Model”, ”Question”, and ”Measure”, as well as the properties ”(Company) uses Model”, ”(Model) consists of (Question)” and ”(Question) has Measure”. Each property has at least one domain concept (e.g. the domain concept for the property ”uses Model” is the concept ”Company”). Its range may either be a literal (e.g. the ”Company-ID” property for the concept ”Company”), or a set of at least one concept (e.g. the range concept for the property ”uses Model” is the concept ”Model”). Domain and range concept restrictions are treated conjunctively. Consequently, all of them must be fulfilled for each property instantiation. Further, it is possible to say that two properties are inverse to each other. Concepts and properties can basically be arranged in a hierarchy. The hierarchy relation relates directly connected concepts (properties) and it is defined as a transitive relationship. The ontology has an instance pool associated with it. It is constructed by specifying instances (e.g. companies c1, c2... cn and maturity models m1, m2... mn) of different concepts and by establishing property instantiations between instances (e.g. ”company c1 uses maturity model m2). Property instantiations must follow the domain and range constraints. In the following two sections, the conceptual layer of our developed KM Maturity Model Ontology is described in detail. The ontology provides the underlying data model for retaining (instantiating) user-defined maturity models, which are including levels, questions and associated measures. Furthermore, the ontology stores authors and users of a maturity model, general information about an organization to be evaluated, which is used for statistical purposes, as well as information about an organization’s current (and past) maturity level(s) and measures to be carried out in order to achieve a higher maturity level.

3.3.2 Retaining KM Maturity Models

An author or maintainer of a maturity model can be instantiated by the ontology using the concept ”Author”. Maturity models created by an author are referenced via the ”(author) hasCreated-Model” property. Additional attributes for the ”Author” concept are ”name” and ”email address”. In order to instantiate different models by the ontology, each model is an instantiation of the ”(Maturity-) Model” concept. A maturity model is specified by the attributes ”model name”, ”start-” and ”stop-level” as well as by the attribute ”count direction”. The ”hasTopic” property describes the topic (e.g. ”KM”, ”software engineering”, etc.) of the maturity model. Valid answers are marked with the ”hasValidAnswer” property. This allows to store maturity models, which use more possible answers than the usual yes/no. The creation date of a maturity model is described by the ”Model-CreationDate” attribute. Every maturity model marks its associated questions with the ”consistsOf” property. The ”Question” concept is linked to a level by the use of the ”hasLevel” property, which indicates to which level a question belongs. The ”hasAnswer” property states whether the right answer of the question instance is ”yes”, ”no” or another user-

defined value. The creation date of a question is stored in the ”Question-Creation-Date” attribute. The actual question is described by the ”Question-Text” attribute, while the ”is-Active” attribute states, whether the question instance should be presented to the user or not. The property ”(question) hasMeasure” indicates the connection between the concept ”Question” and the Concept ”Measure”, whose attribute ”Measure-Text” is displayed to the evaluating organization, if that question has been answered wrong. In addition, the ”hasLink” property points to the ”Link” concept which comprises a ”Link-URL” and a ”Link-Description” attribute. The ”hasImage” property is related to the ”Link” property and stores an image URL as well as an image description. The ”Saved-State” concept contains the history of selected changes to the ontology. Every time a company finishes answering questions of a level, relations are instantiated to already answered and accomplished questions for statistical purposes. In order to realize the demand for retaining maturity models, that are a subset of (an) already defined maturity model(s), the KM Maturity Model Ontology provides the property ”(model) consists of model”. Furthermore an inverse property ”(model) is part of model” has been defined. Questions and levels that have to be arranged in a hierarchy are treated analogously by the use of the properties ”(level) consists of level” and ”(question) consists of question” and their corresponding inverse properties ”(level) is part of level” and ”(question) is part of question. Due to a more concise view, part-of-relations are faded out in figure 3.2. Figure 3.3 shows the concept ”model”, including these previously faded out properties ”(model) consists of model” and ”(model) is part of model”. These properties are used to unitize/subdivide models or to derive new models from existing ones. Alternatively, it is possible to create (meta-) models that are a superior model of (an) already existing model(s). Furthermore, the hierarchization of models (and with it the hierarchization of questions and methods) provide the distributed creation of (complex) models by several authors as well as the distributed self-evaluation by several persons of an organization.

Figure 3.3: Hierarchization of models

3.3.3 Identification of a Company’s Maturity Level

Each registered company is represented as an instance of the ”Company” concept that comprises properties for company size, sector and location as well as the attributes ”Login-Counter”, ”Registration-Date”, ”Last-Login-Date” and ”Company-ID” that are used for statistical purposes. The property ”hasAnswered” marks already answered questions of a company, while the property ”hasAccomplished” marks questions that were an-

swered correctly. Defined measures are later determined by looking for questions that are marked with an instantiation of the property ”hasAnswered” but not with an instantiation of the property ”hasAccomplished”. The property ”hasCur-rentQuestion” is used to mark questions for temporary internal purposes of the system. In order to verify, if a company has reached a particular maturity level, the later described evaluation component accesses the ontology and checks out, if all questions of that level have been answered right. This will be done by checking the availability of the property instantiation ”hasAccomplished” for every answered question. If there exists a question that has not been answered right by the user, the corresponding instance of the concept ”Measure” is presented to the user, which is directly related to the wrong answered question using an instantiation of the property ”(Question) has Measure”. After the realization of the provided measure(s) and the consecutive system login, the system again provides previously wrong answered questions to the user. If the company now answers right all questions and with it reached a particular maturity level, the level is stored into the ontology by a new drawn instantiation of the property ”hasCompanyLevel”. To identify the state of evaluation for a company that would like to continue a not yet completed evaluation on a particular level, the ontology provides the property ”hasAnswered”. This will guarantee that already answered questions of a particular level are not given again to the user.

3.4 Related Work

At this stage, there exist several theoretical but also tool-supported models for identifying the current KM maturity level of an organization (i.e. the KM Maturity Model (KMMM®) from Siemens [LE04], Kochikar’s KM Maturity Model [Koc00], Berztiss’ Capability Maturity for KM [Ber02], the Knowledge Process Quality Model (KPQM) [PP02] and others) which base in the majority of cases on the Capability Maturity Model® for Software (SW-CMM®) [PWCC95]. An overview and detailed description of KM maturity models is also given in [WPH02]. However, from our point of view, our tool-supported approach of an easy maintainable and therefore sustainable ontology-based software infrastructure for web-based self-evaluation is rather unique in this context.

3.5 Conclusion and Future Work

We have described ONTOKNOM^3, an ontology-based software infrastructure for technically supporting the retainment, representation and autonomous application of any theoretical KM maturity model. This will be realized by providing a KM Maturity Model Ontology as well as a tool kit for supporting the creation, modification and enhancement of different maturity models or even their synergetic combination. Based on one or more selected or even nestable maturity model(s), ONTOKNOM^3 furthermore supports the web-based identification of an organization’s KM maturity level and provides based

on the underlying model(s) organizational measures to achieve a higher one. In the future we will perform the further validation of ONTOKNOM^3 by integrating and combining additional existing theoretical KM maturity models. Furthermore we will evaluate the capability of our approach on providing means for the flexible management of ontology-based maturity models as well as their flexible combination. This will be done by e.g. applying ONTOKNOM^3 for consulting organizations in the holistic introduction of KM more efficiently. Moreover, critical, time- or cost-consuming steps of a KM implementation, that require the optimization or even replacement of recommendations and measures will be identified by the use of ONTOKNOM^3.

Chapter 4

DILIGENT Process and Argumentation Framework

We here sketch the DILIGENT process and the argumentation framework. In Section 4.1 we motivate our work with a typical scenario for our approach to develop ontologies. Following in Section 4.2 is an overview of the DILIGENT process. The process is refined in two ways. On the one hand we offer a detailed action list, adapted for a particular use case, in Section 4.3. On the other hand we analyzed the process w.r.t. other engineering methodologies, and provide a very general detailed description in Section 4.4. The process has already been applied in a real world scenario. Results and lessons learned from the application are described in Section 7.4. Finally, we elaborate on a thorough analysis of a taxonomy evolution in the biology domain. The analysis motivated our extension of the DILIGENT process by an argumentation framework and the development of an argumentation ontology. The hypothesis generated from the analysis was evaluated in two in-situ experiments at the Institute AIFB. The analysis as well as the experiments are described in Section 4.6.

4.1 Motivational Scenario

In *distributed* development there are several experts, with different and complementary skills, involved in collaboratively building the same ontology. For instance, in Virtual Organizations, Open Source and Standardization efforts, experts belong to different competing organizations and are geographically dispersed. In these cases, builders typically are also users and, although some users are not directly involved in changing the ontology, they take part in the process by using the ontology.

An initial ontology is made available and users are free to use it and modify it locally for their own purposes. There is a central board that maintains and assures the quality of the shared ontology. This central board is also responsible for deciding updates, but these

are based on user re-occurring changes and requests, therefore the board *loosely controls* the process. It is expected that the change rate of the ontology made available should be higher than the usual due to maintenance, therefore this is a more *evolving* process.

4.2 DILIGENT Overview

We will now describe the general process, roles and functions in the DILIGENT process. As shown in Figure 4.1 it comprises five main activities: (1) **build**, (2) **local adaptation**, (3) **analysis**, (4) **revision**, (5) **local update** (*cf.* figure 4.1). The process starts by having *domain experts*,*users*, *knowledge engineers* and *ontology engineers* **build**ing an initial ontology. In contrast to known ontology engineering methodologies available in the literature [GPS98, GPFLC03, PM01, UK95] our focus is distributed ontology development involving different stakeholders, who have different purposes and needs and who usually are not at the same location. Therefore, they require online ontology engineering support.

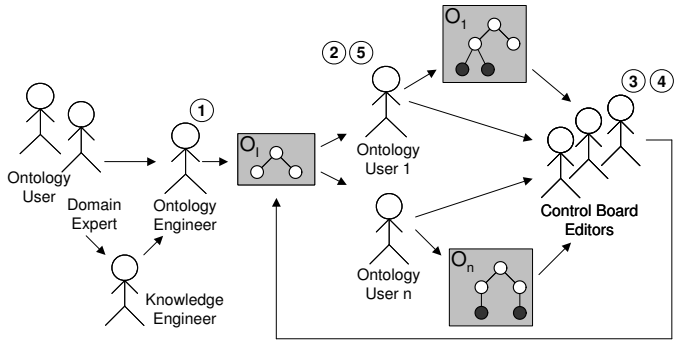

Figure 4.1: Roles and functions in distributed ontology engineering

The team involved in building the initial ontology should be relatively small, in order to more easily find a small and consensual first version of the shared ontology. Moreover, we do not require completeness of the initial shared ontology with respect to the domain.

Once the product is made available, users can start using it and **locally adapting** it for their own purposes. Typically, due to new business requirements, or user and organization changes, their local ontologies evolve in a similar way as folder hierarchies in a file system. In their local environment they are free to change the reused shared ontology. However, they are not allowed to directly change the ontology shared by all users. Furthermore, the control board collects change requests to the shared ontology.

The board **analyses** the local ontologies and the requests and tries to identify similarities in users' ontologies. Since not all of the changes introduced or requested by the users will be introduced,[1] a crucial activity of the board is deciding which changes are

[1]The idea in this kind of development is not to merge all user ontologies.

going to be introduced in the next version of the shared ontology. The input from users provides the necessary arguments to underline change requests. A balanced decision that takes into account the different needs of the users and meets user's evolving requirements for the ontology[2] has to be found. The board should regularly **revise** the shared ontology, so that local ontologies do not diverge too far from the shared ontology. Therefore, the board should have a well-balanced and representative participation of the different kinds of participants involved in the process.

In this case, users are involved in ontology development, at least through their requests and re-occurring improvements and by evaluating it, mostly from an usability point of view. Knowledge providers in the board are responsible for evaluating the ontology, mostly from a technical and domain point of view. Ontology engineers are one of the major players in the analysis of arguments and in balancing them from a technical point of view. Another possible task for the controlling board, that may not always be a requirement, is to assure some compatibility with previous versions. Revision can be regarded as a kind of ontology development guided by a carefully balanced subset of evolving user driven requirements for the ontology. Ontology engineers are responsible for updating the ontology, based on the decisions of the board. Revision of the shared ontology entails its evolution.

Once a new version of the shared ontology is released, users can **update** their own **local** ontologies to better use the knowledge represented in the new version. Even if the differences are small, users may rather reuse *e.g.* the new concepts instead of using their previously locally defined concepts that correspond to the new concepts represented in the new version.

4.3 Detailed Process Description: Use Case Oriented

In order to provide a detailed guidance for the participants in the process and to identify potential technical support for the single process steps we have analysed them with more detail. The result is depicted in figure

The analysis includes the identification of (1) the major roles, (2) the input and (3) output information, (4) the decisions and (5) the actions within the process step.

We provide a detailed description of the process on two levels. The description provided in this section focuses on concrete actions a user has to perform to reach the desired output. In the next section we look at the process from an ontology engineering perspective and elaborate on the activities an ontology engineer follows, when applying the process model.

[2]This is actually one of the trends in modern software engineering methodologies (see Rational Unified Process).

4.3.1 Local Adaptation: Detailed View

1. **Roles** The actors involved in the local adaptation step are users of the ontology. They use the ontology to retrieve e.g. documents which are related to certain topics modelled in the ontology or more structured data like the projects an employee was involved in. Information gathering need not be their main objective, but they may rather need the information to fulfill their individual tasks.

2. **Input** Besides the common shared ontology, in the local adaptation step the information available in the local information space is used. This can be existing databases, ontologies or folder structures and documents.

3. **Output** The output of the process step is a locally changed ontology which better reflects the users needs. Each change is supported by arguments explaining the reasons for a certain change. We here emphasize that changes are not propagated to the shared ontology. Only in the *analysis step* the board gathers all ontology change requests and the corresponding arguments to be able to evolve the common shared ontology in the *revision step*. The result is a set of locally changed ontologies and a set of requests for changes made to the board. All these are associated to arguments underlying the required changes.

4. **Decisions** The actors must decide which changes they want to make to their on-

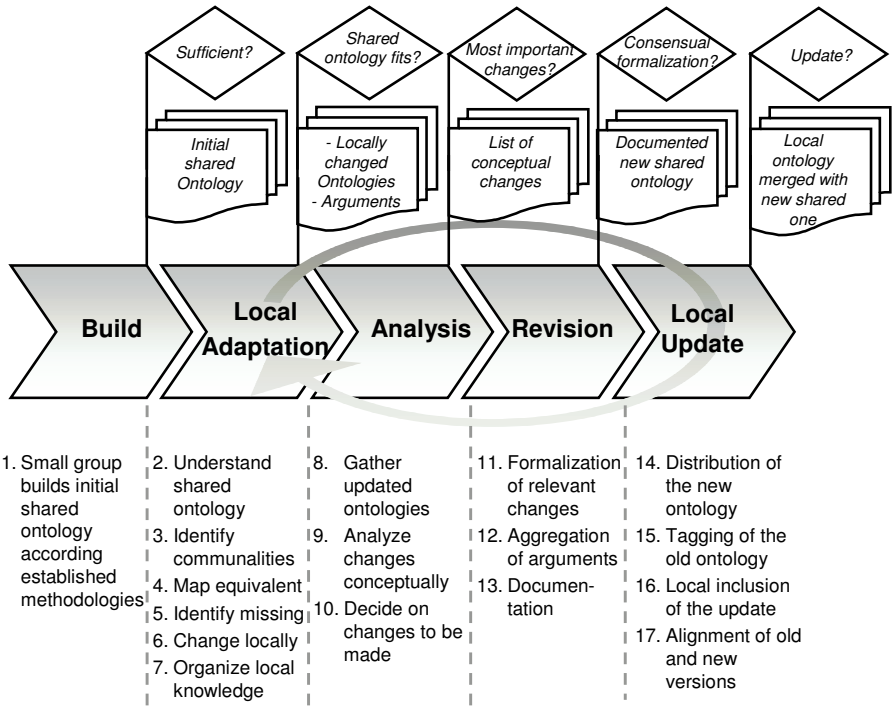

Figure 4.2: Process stages (1-5), actions (1-17) and structures

tology. Hence, they must decide if and where new concepts are needed and which relations a concept should have. [3]. They must further provide reasons why they made certain decisions. To evaluate the decisions we propose to calculate the ratio between available information and the information which can be classified according to the adapted ontology. The proportion should ideally be high. Further classifications should be specific. Local concepts which could not be aligned with shared concepts should be introduced as local adaptations.

5. **Actions:** To achieve the desired output the user takes different actions namely *Understand shared ontology*, *Identify commonalities between own and shared conceptualization*, *Map equivalent conceptualizations of different actors*, *Identify missing conceptualizations*, *Change conceptualization* and finally *Organize local knowledge according to the conceptualization*.

   The last three actions of the process are performed in a cyclic manner until a new common ontology is available and the entire process step starts again.

   The single actions performed manually would require a grounded understanding of ontologies and their underlying formal representation. We cannot expect such knowledge from all actors participating in the process. The process should rather be integrated seamlessly in the environment the user works in. Hence we now indicate for each of the actions the available technology to support the actors.

6. **Tool support:** Building is supported by existing ontology editors like [SEA+02b, NFM00]. In [LET04] we describe how existing structure on local machines can be utilized to facilitate the creation of ontologies. The tool supports thus actions (3) and (5). We have further integrated ontology mapping to support step (4) [ES04]. (6) is a manual step. (7) is currently a manual step, too, but it could be supported by semi automatic classification *cf. e.g.* [HSC02].

- **Understand shared ontology:** An ontology is a conceptualization of the real world. An ontology should represent a shared conceptualization. In fact a completely shared ontology can never be engineered, since different people have varying interpretations of the real world. Therefore it is necessary as a first action to relate the own interpretation of the world to the shared conceptual model. Thus the actor must learn where the different concepts are located in the ontology and how they are interrelated with other concepts. The ontology can be very complex, thus comprehension of the ontology depends mainly on its presentation. Different technologies can be used to provide the user with a context sensitive view on the ontology which does not overwhelm him. Relevant technology to support this actions are text classification methods, natural language processing and ontology learning methods. We might also consider alternatives to technology driven teaching methods, *e.g.* handbooks or offering a new tip to users each day.

[3] When we talk about changing the ontology or introduce new concepts this applies also to relations and axioms

- **Identify commonalities between own and shared conceptualization:** Following the comprehension of the ontology the user can realize the communality between the own and shared conceptualization. We here point to the work of [SG89] which we introduced in section 2.5.1.4. He identified the different types of conflict when comparing two or more ontologies.

  To support this step technically we can use the available formal conceptualization on the local machines. To identify the degree of communality we can use mapping methods to find correspondences between locally available formal models and the shared ontology.

  The documents can be operationalized in part for ontology learning which then identifies concepts and relations based on the local text or the documents the user has browsed through.

- **Map equivalent conceptualizations of different actors:** After the identification of commonalities it is necessary to make them explicit. Otherwise the system will not be able to make use of the findings. The expressivity of the used ontology language may set a limit this. For example explicit formalization of mappings is only possible with OWL. RDF(S) does not support it originally. Different implementations may add specialized add-ons. Mappings have the advantage, that they leave the original structures unchanged. Of course users may also decide to change their local structures in favour of the common structure. In this case the changes must be traceable, so that the actor can retain its old version.

- **Identify missing conceptualizations:** Besides the identification of communality the same techniques can be applied in the subsequent step to support the user in identifying missing conceptualizations.

  Depending on the scenario the user might have access to other users ontologies and use their local adaptations as further input to identify missing concepts in her own conceptual model.

- **Add missing conceptualizations:** After identification of missing conceptualizations the user must be enabled to introduce the changes. This is not so much a technical challenge than a usability challenge. The user should not be bother with suggestions all the time and he might not tolerate wrong suggestion. Since automated methods can not be 100% correct it depends on the user context when and how to apply the changes to the ontology.

  The board analysis the changes performed by the users. To be able to understand the change requests the actor should provide reasons for each request. Again the rationales used within the automated methods can be used as input here. To support the user further in providing reasons a part of the process model is an argumentation framework to focus the user on the relevant arguments he can provide.

- **Organize local knowledge according to ontology:** At this point the ontology should reflect the users conceptualizations. Now he can instantiate the ontology with the information available locally and hence contribute to the collective knowledge. Again text classification and natural language processing can be used to facilitate this action.

The implementation of tools to support the single actions must be done in close cooperation with the user and with respect to usability. The steps are complex so that an easy way must be found to enable the users to follow the process.

We have shown how different techniques developed in the course of the SEKT project can be used within the process to support the actors to follow the process. The output is a locally adapted ontology. Hence the board can retrieve the changes and analyse the reasons underlying each change. The reasons can either provided automatically by the supporting methods or manually following the argumentation model describe with more detail in section 4.6.

4.3.2 Analysis

Depending on the frequency and volume of changes the board will make adjustment cycles as needed.

1. **Roles** In the analysis stage we can distinguish three roles played by board members: (i) The domain expert decides which changes to the common ontology are relevant for the domain and which are relevant for smaller communities only. (ii) Representatives of the users explain different requirements for the shared ontology from the usability perspective. At this stage, work is conducted at a conceptual level. (iii) The ontology engineers analyze the proposed changes from a knowledge representation point of view foreseeing whether the requested changes could later be formalized and implemented.[4]

2. **Input** The analysis stage takes as input the ontology changes proposed and/or made by the participating actors. To be able to understand the change requests, users should provide their reasons for each request. Both manual and automated methods can be used in the previous stages. Besides of arguments by ontology stakeholders, one may here consider rationales generated by automated methods, e.g. ontology learning. The arguments underlying the proposed changes constitute important input for the board to achieve a well balanced decision about which changes to adopt.

3. **Output** The result is a list of the major changes to be introduced that were agreed by the board. Hence, all changes which should not be introduced into the shared ontology are filtered. In this stage it is not required to decide the final modelling of the shared ontology.

[4] In the revision stage.

4. **Decisions** The board must decide which changes to introduce into the new shared ontology at the conceptual level. Metrics to support this decision are (i) the number of users who introduced a change in proportion to all users who made changes. (ii) The number of queries including certain concepts. (iii) The number of concepts adapted by the users from previous rounds.

5. **Tool support:** In [PSST04] we present an extension to an ontology editor, which supports actions (8) and (9) and (10). (8) Ontologies can be collected from the users in a Peer-to-Peer system. Different sorting and grouping mechanisms help the board to analyze the introduced changes systematically. The identification of relevant changes is in the end a community process. Here we support decision making by structured argumentation support as described in [TPSS05].

6. **Actions:** To achieve the desired output the board takes different actions namely *Gather locally updated ontologies and corresponding arguments*, *Analyse the introduced changes* and *Identify changes relevant for all actors*.

- **Gather locally updated ontologies and corresponding arguments:** Depending on the deployed application the gathering of the locally updated ontologies can be more or less difficult. It is important that the board has access to the local changes to be able to analyse them. In a centralized ontology based system in which users can make their changes within their workspace this task is very easy. In peer-to-peer scenarios some peers might not always be reachable, to be able to collect the local ontologies. For the board it might also be interesting not only to analyse the final changed ontology, but also the evolution process. However, with an increasing number of participants this in-depth analysis might not be feasible. Since analysis takes place at the conceptual level, reverse engineering is usually an important technique to get the conceptual model from the formalized model [GPFLC03]. To support users providing their reasons, an argumentation framework that focuses the user on the relevant arguments was developed *cf.* [TPSS05].

- **Analyse the introduced changes:** The number of change requests may be huge and also contradictory. First the board must identify the different areas in which changes took place. Within analysis the board should bear in mind that changes of concepts should be analyzed before changes of relations and these before changes of axioms. Good indicators for changes relevant to the users are (i) overlapping changes and (ii) their frequency. Furthermore, the board should analyze (iii) the queries made to the ontology. This should help to find out which parts of the ontology are more often used. Since actors instantiate the ontology locally, (iv) the number of instances for the different proposed changes can also be used to determine the relevance of certain adaptations.

- **Identify changes relevant for all actors:** Having analyzed the changes and having grouped them according to the different parts of the ontology they belong to, the

board has to identify the most relevant changes. Based on the provided arguments the board must decide which changes should be introduced. Depending on the quality of the arguments the board itself might argue about different changes. For instance, the board may decide to introduce a new concept that better abstracts several specific concepts introduced by users, and connect it to the several specific ones. Therefore, the final decisions entail some form of evaluation from a domain and a usage point of view. The outcome of this action must be a reduced and structured list of changes that are to be accomplished in the shared ontology.

4.3.3 Revision

While we could evaluate the local update and analysis step of our process model already in small experiments the revision phase and local update phase are not well tested yet. Hence, due to the early stage of the project the revision phase and the local update phase are not yet very well elaborated. We here just sketch some general observations and point our future directions of research.

1. **Roles** The ontology engineer judges the changes from an ontological perspective, more exactly at a formalization level. Some changes may be relevant for the common ontology, but may not be correctly formulated by the users. The domain experts should judge and decide whether new concepts/relations should be introduced into the common ontology even so they were not requested by the users.

2. **Input** The input for the revision phase is a list of changes at a conceptual level which should be included into the ontology.

3. **Decisions** The main decisions in the revision phase are formal ones. All intended changes identified during the analysis phase should be included into the common ontology. In the revision phase the ontology engineer decides how the requested changes should be formalized. Evaluation of the decisions is performed by comparing the changes on conceptual level with the final formal decisions. The differences between the original formalization by the users and the final formalization in the shared ontology should be minimal.

4. **Actions:** To achieve the desired output the user takes different actions namely (11) *Formalization of the requested changes*, (12) *Aggregation of arguments* and (13) *Documentation*. Judging entails *Evaluation* of proposed changes from a knowledge representation/ontological point of view.

5. **Output** The revision phase ends when all changes are formalized and well documented in the common ontology.

6. **Tool support:** For the revision phase we do not envision any special tool support beyond the one provided by classical ontology engineering environments. In particular the ontology evaluation framework described in the companion deliverable 7.2.1 will help the users in this activity.

- **Formalization of the requested changes:** Similar to established methodologies the requested changes must be formalized with respect to the expressivity of the ontology. We will not go into detail with this step since it is already described in methodologies referred to in the related work section.

- **Aggregation of arguments:** As arguments play a major role in the decision process we expect that the changes which are eventually included into the common ontology are supported by many arguments. One of the reasons for keeping track of the arguments is to enable users to better understand why certain decisions have been made with respect to the ontology. Hence, the user should be able to retrieve the most convincing arguments made to introduce a certain change.

- **Documentation** With the help of the arguments, the introduced changes are already well documented. However, we assume that some arguments might only be understandable for the domain expert and not for the users. Hence, we expect that the changes should be document to a certain level.

4.3.4 Local Update

1. **Roles** The local update phase involves only the users. They perform different actions to include the new common ontology into their local system before they start a new round of local adaptation.

2. **Input** The formalized ontology including the most relevant change request is the input for this step. We also require as an input the documentation of the changes. For a better understanding the user can request a delta to the original version.

3. **Output** The output of the local update phase is an updated local ontology which includes all changes made to the common ontology. However, we do not require the users to perform all changes proposed by the board. The output is not mandatory, since the actors could change the new ontology back to the old one in the local adaptation stage.

4. **Decisions** The user must decide which changes he will introduce locally. This depends on the differences between the own and the new shared conceptualization. The user does not need to update her entire ontology. This stage interferes a lot with the next local adaptation stage.

5. **Actions:** To achieve the desired output the user takes different actions namely (14) *Distribution of the new ontology to all actors*, (15) *Tagging of the old ontology* to allow for a roll back, (16) *Local inclusion of the updated version* and (17) *Alignment of old and new versions*.

6. **Tool support:** The Local update stage is very critical from a usability point of view. Changes cannot be introduced without the user's agreement. Further he should not be bothered too often. In case of equivalent but local conceptualizations it must be possible to change to the common conceptualization. From a technical point of view this stage is supported by tools like KAON *cf.* [MMS03]. We do not exclude the possibility of conflicts. Unresolved conflicts, though, will reduce the utility of the ontology. Even in this cases attempts are under way to resolve them as far as possible automatically *cf.* [HvHH+05].

- **Distribution of the ontology to all actors:** Analogously to step 4.4.3 the shared ontology must be distributed to the different participants. Depending on the overall system architecture different methods can be applied here.
- **Tagging of the updated version:** To ensure user satisfaction, the system must enable the user to return to her old version of the ontology at any time. The user might realize that the new updated version of the common ontology does not represent her needs anymore and thus want to leave the update cycle out. To reach a better acceptance this must be possible and is foreseen in the methodology. The user can always balance between the advantages of using a shared ontology or using her own conceptual model.
- **Inclusion of the updated version:** The system must support the user to easily integrate the new version into her local system. It must be guaranteed that all annotations made for the old version of the ontology are available in the new version.
- **Update of local adaptations which are not included in the common ontology:** The update of the local ontology can lead to different kinds of conflict. Changes proposed by the user may indeed have found their way into the common ontology. Hence, the user should be enabled to use from now on the shared model instead of her own identical model. Furthermore, the board might have included a change based on arguments the user was bringing forward, but has drawn different conclusions. Here the user can decide whether he prefers the shared interpretation. Other option might emerge in the course of the case studies.

4.4 Generic Detailed Process Description

In this section we describe the details of the DILIGENT process on a generic level. Instead of describing the actions a particular user has to perform in a specific use case, we

provide a generic activity description; this, however, must be adapted, if it is applied in a specific use case. The generic description is inspired by the mapping between the cost model (*cf.* chapter 5) and the DILIGENT process. The mapping revealed some problems w.r.t. the use case specific definition of the process actions (Section 5.3.1). The process actions are not defined at the same conceptual or granularity level from an ontology engineering perspective. The cost model shall be applicable independently of the chosen ontology engineering process, thus its definition must be generic[5]. Therefore, we decided to generalize the actions defined so far, and integrate them into more general activities. The process stages and the the cost factors can thus easily be aligned (*cf.* 5.3). The generic detailed process model is depicted in figure 4.3

As in the use case oriented description, we have analyzed the different process stages in detail. For each stage we have identified (i) major roles, (ii) input, (iii) decisions, (iv) activities (v) and output information that occur in each stage. We do not define tool support in this description, as this is use case specific. In some cases the general description an the use case specific one is equivalent; we repeat the description in these cases.

4.4.1 Building

The build phase in the DILIGENTprocess model is so far not defined, as we assume that this step is performed according to established engineering methodologies. In order to achieve a homogeneous description of the process activities from a granularity point of view we decided to further specify this phase at the same level of detail as the subsequent ones. As summarized in [GPFLC03] the available methodologies separate the building process in different activities, while each of the methodologies focuses on different aspects. For our purposes and for the alignment with the cost model we separated ontology building into the four activities *Domain analysis*, *Conceptualization and implementation of shared ontology*, *Evaluation of shared ontology*, *Argument provision* and *Documentation*.

In contrast to a common full ontology engineering cycle the objective of this Build task is not to generate a complete and evaluated ontology but rather to *quickly* identify and formalize the main concepts and main relations.

4.4.1.1 Roles

Classical ontology engineering methodologies introduce three different roles: Knowledge engineer, ontology engineer and domain expert. The domain expert provides the knowledge engineer and ontology engineer with the domain knowledge to be modeled. The domain expert knows for the domain of interest the relevant concepts and the interdependencies between them; he can point to further information found in the literature. The

[5]We introduced new and redefined cost factors as well in the cost model. This is reported in section 5.3.2

knowledge engineer extracts from the domain expert the conceptual model w.r.t. to the domain. The ontology engineer generates from the conceptual model a machine readable ontology. It is often observed that the same person takes the role of the knowledge engineer and ontology engineer.

Additionally to these classical roles we also propose the involvement of users in this stage. The user should evaluate the designed ontology from a usability perspective.

The persons involved in the build stage are the initial board members.

4.4.1.2 Input

The build stage takes as input the task to build an ontology and the results of a feasibility study. This includes a general task description and the identification of main domain experts.

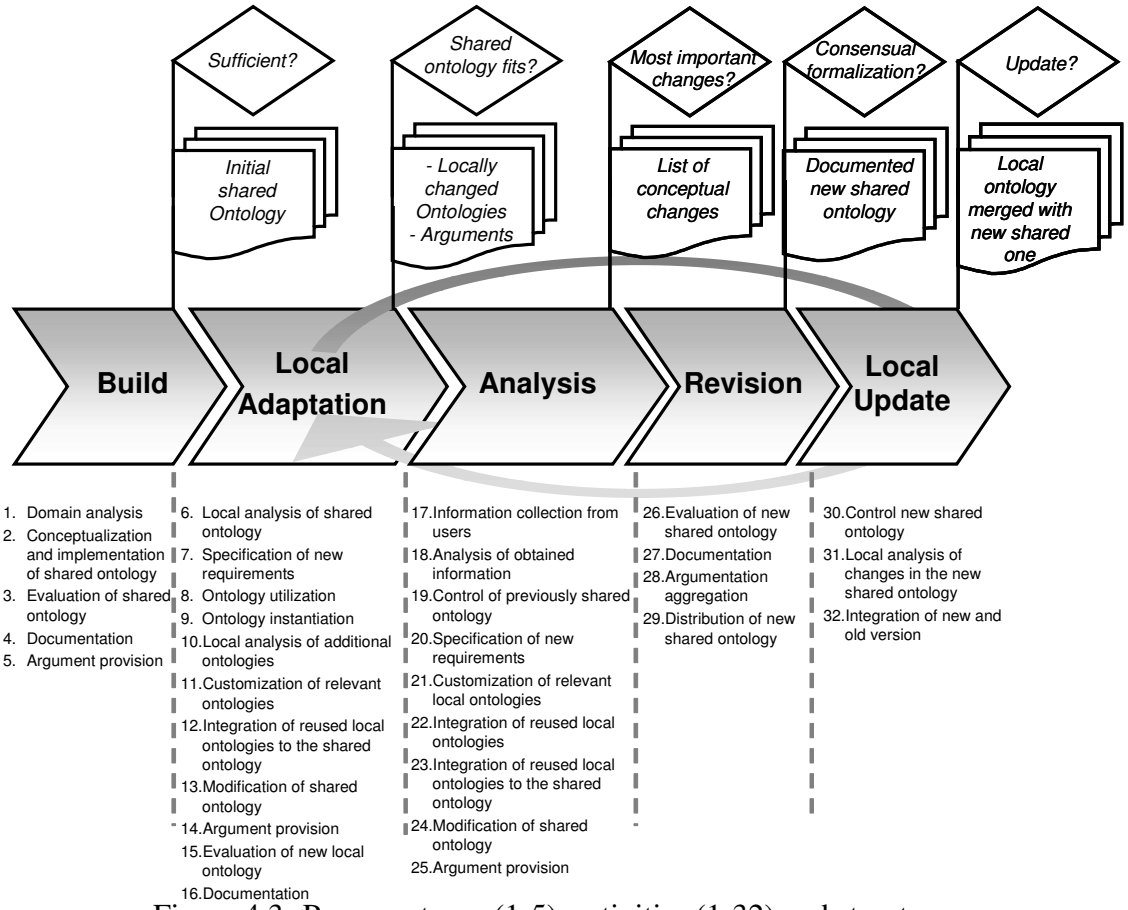

Figure 4.3: Process stages (1-5), activities (1-32) and structures

4.4.1.3 Output

The result of the build stage is an ontology, which models the main concepts of the domain. In contrast to established methodologies we do not require completeness of the domain model.

4.4.1.4 Decisions

Critical decision's of the build stage are: How big shall the initial ontology be? Does the initial ontology capture sufficient domain knowledge? Can the users understand the initial ontology?

4.4.1.5 Activities

In the following we describe the activities of the central build stage, namely *Domain analysis*, *Conceptualization and implementation of shared ontology*, *Evaluation of shared ontology*, *Argument provision* and *Documentation*.

Domain analysis In our case the domain analysis covers all tasks necessary to conceptualize the ontology. We thereby assume that the result of the feasibility study for the project was positive. A detailed definition on the execution of a feasibility study can be found in the OTK methodology [SAB+03, SS02]. The domain analysis itself covers tasks as defined in the specification phase in METHONTOLOGY [GPFLC03]. In the OTK methodology this activity is performed in the kickoff phase. We emphasize that the domain analysis includes the definition of modules covering sub domains in order to keep the building trackable. Competency questions are a good possibility to capture detailed requirements for the ontology. An interesting special case occurs when existing ontologies should be reused. Here, domain analysis involves also the identification and selection of the reusable ontologies [PM00]. METHONTOLOGY introduces also management activities supporting the building process. These are of course necessary during the entire building process. However, as our process explicitly assumes only loose control we leave the level of management to the participants and their particular requirements for overall management. Knowledge acquisition is foreseen as a parallel activity to the building process in METHONTOLOGY. We acknowledge that knowledge acquisition is performed in all stage of the process but rather regard it as an implicit activity than an explicit one.

Conceptualization and implementation of shared ontology In contrast to other methodologies we subsume conceptualization and implementation of the ontology as one activity. As most building processes are supported by ontology editors, the actual implementation is mostly done automatically. For the applicant of the methodology the

separation seems thus artificial. Conceptualization and implementation covers the tasks essential to build the shared ontology. We here emphasize that the resulting model need not to be complete w.r.t. the domain. The ontology should represent as much detail as necessary for its initial usability and usefulness. However, the costs to build the initial ontology should not outweigh its benefits.

Argument provision In existing ontology building methodologies the capturing of the argumentation during the construction was neglected. In the field of Software Engineering capturing of arguments during the requirements analysis has been well-researched in the last decade. We have introduced an argumentation model tailored for the specific requirements for ontology building and maintenance [TPSS05] (*cf.* section 4.7). In our process the explanation of the design principles underlying the conceptual model is of particular interest, since not all users are necessarily involved in the first building process but are allowed to change the ontology in later stages. Therefore, the arguments in favor and against certain design decisions must be captured to allow the users of the ontology to extend and change the shared ontology adequately.

Evaluation of shared ontology Ontology evaluation is not a mature area of ontology engineering yet. Several approaches to evaluate ontologies propose partial solutions for the evaluation of ontologies, from a general-purpose or a usage-related perspective. Approaches to evaluate ontologies in the first category introduce methods focusing on the ontologies schema, i.e. the quality of the conceptual model [GPFLC03, GW02, UHW+98]. Assessing the usability of an ontology in a target application context, the second category, is addressed for example in OntoMetric [LTGP04], a framework for selecting appropriate ontologies. Additionally to these categories, we find approaches to evaluate ontologies aiming at evaluating the a-posteriori usage of an ontology for a specific task such as semantic annotation of texts.

Documentation Ideally the documentation step should be performed in parallel to the aforementioned activities, in order to ensure an efficient process quality control and improve the reusability of the prototypical ontology, which is subject of further changes on the basis on its usage in a variety of environments in the distributed setting. Likewise Software Engineering it is crucial that the documentation is performed not only at the implementation level, but during the entire building process, including domain analysis and evaluation. Additionally to standard documentation procedures, in our process it is helpful to capture the participants in the original building process, who were involved in the initial design decisions.

4.4.2 Local Adaptation

Once an initial ontology is *built* and released, users will start to adapt it locally for their own purposes.
New shared ontologies are made available to users either by push or pull mechanisms. The users deciding to utilize a new version of the shared ontology first get familiar with the shared ontology in order to be able to use it correctly. In the next step they may interact with the ontology in parallel in a threefold manner, depending on the concrete application setting. Some users will use the ontology only for retrieving information either locally or from other participants. Others will also actively instantiate the ontology with their own information. Performing both activities the user may detect missing conceptualizations in the shared ontology. This and the analysis of shared ontology can result in the local definition of new requirements for the shared ontology.

4.4.2.1 Roles

The actors involved in the local adaptation step are users of the ontology. They use the ontology to retrieve e.g. documents which are related to certain topics modeled in the ontology or more structured data like the projects an employee was involved in. Information gathering need not be their main objective, but they may rather need the information to fulfill their individual tasks.

4.4.2.2 Input

Besides the common shared ontology, in the local adaptation step the information available in the local information space is used. This can be existing databases, ontologies or folder structures and documents. Furthermore the user can consult at any time the ontologies and e.g. folder structures of other users.

4.4.2.3 Output

The output of the process step is a locally changed ontology which better reflects the users needs. Each change is supported by arguments explaining the reasons for a certain change. We here emphasize that changes are not propagated to the shared ontology. Only in the *analysis step* the board gathers all ontology change requests and the corresponding arguments to be able to evolve the common shared ontology in the *revision step*. The result is a set of locally changed ontologies and a set of requests for changes made to the board. All these are associated to arguments underlying the required changes.

4.4.2.4 Decisions

The actors must decide which changes they want to make to their ontology. Hence, they must decide if and where new concepts are needed and which relations a concept should have. [6]. They must further provide reasons why they made certain decisions. To evaluate the decisions we propose to calculate the ratio between available information and the information which can be classified according to the adapted ontology. The proportion should ideally be high. Further classifications should be specific. Local concepts which could not be aligned with shared concepts should be introduced as local adaptations.

4.4.2.5 Activities

Conceptualizing the new requirements for the ontology incorporates activities as they are known from classical ontology engineering. Ontology users might decide to integrate existing ontologies (originating from different local parties involved in the process) to the local one or to conceptualize the desired changes from scratch. As other users, in particular the board, should be able to understand these changes, argument provision becomes crucial. Documentation is the last activity in the process. We acknowledge though that documentation was rarely done in most of our case studies. Figure 4.4 visualizes the dependencies between the single activities.

To achieve the desired output the user takes different activities namely *Local analysis of shared ontology*, *Specification of new requirements*, *Ontology utilization*, *Ontology instantiation*, *Local analysis of additional (local) ontologies*, *Customization of relevant (local) ontologies*, *Integration of reused (local) ontologies to the shared ontology*, *Modification of shared ontology*, *Argument provision*, *Evaluation of new local ontology* and *Documentation*.

The evaluation of the adapted shared ontology might suggest that not all local requirements for the ontology are met yet, thus we define a cyclic behavior inside the process stage. The single activities performed manually would require a grounded understanding of ontologies and their underlying formal representation. We cannot expect such knowledge from all actors participating in the process. The process should rather be integrated seamlessly in the environment the user works in.

Local analysis of shared ontology The goal of this activity is to understand the ontology. An ontology is a conceptualization of the real world. It should furthermore be the result of a common agreement w.r.t. modeling issues. A completely shared ontology can never be engineered, since different people have varying interpretations of the real world. Therefore it is necessary as a first action to relate the own interpretation of the world to the shared conceptual model. Thus the user should learn where the different concepts are

[6] When we talk about changing the ontology or introduce new concepts this applies also to relations and axioms

located in the ontology and how they are interrelated with other concepts. The ontology can be very complex, thus comprehension of the ontology depends mainly on its presentation. Different technologies can be used to provide the user with a context-sensitive view on the ontology to reduce complexity.

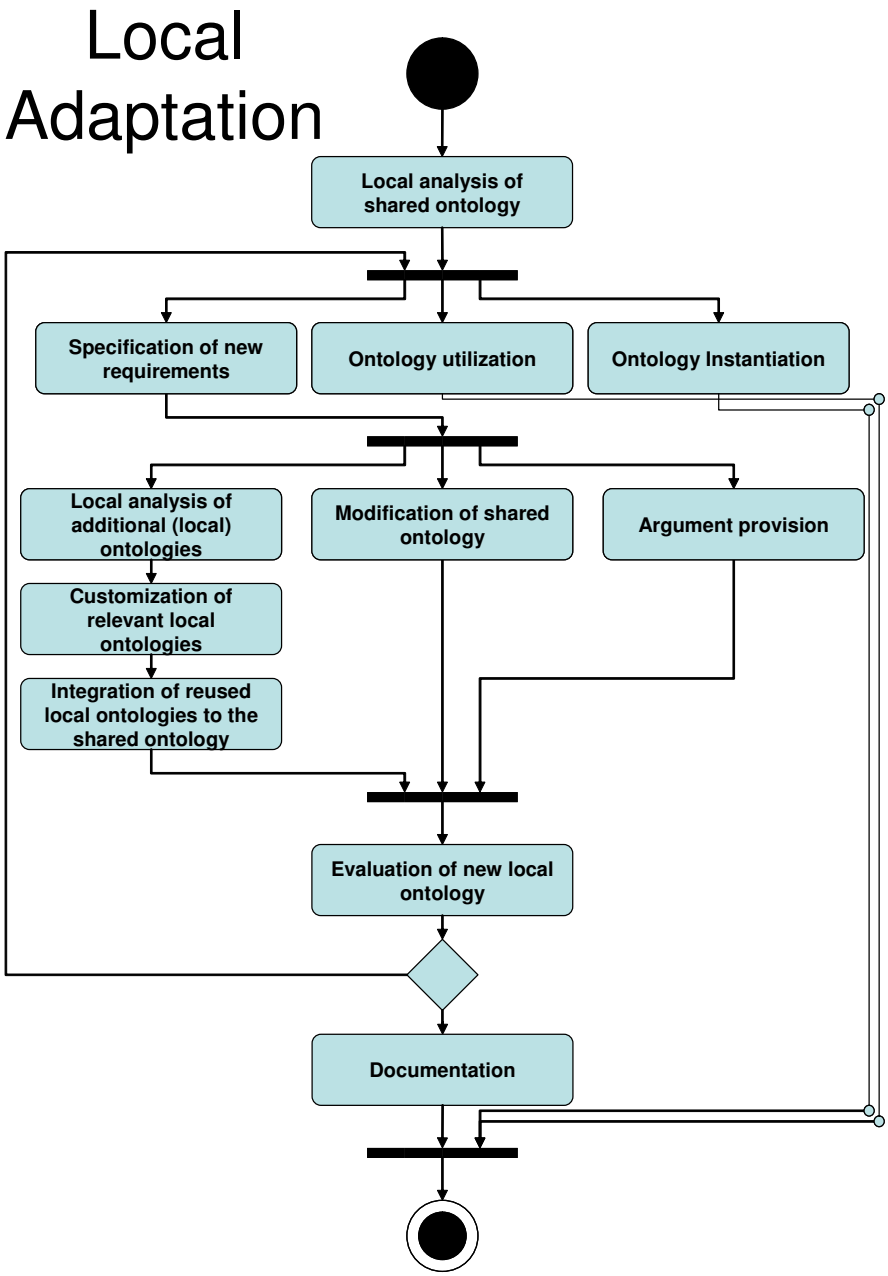

Figure 4.4: Local Adaptation: Activity Diagram

After completing this activity the user is able to instantiate the ontology and to use it to answer her information needs. It is not necessary that the user understands the entire ontology immediately. The analysis can be performed gradually. However for the parts the user modifies, instantiates or uses in subsequent activities she must first understand what these parts are about.

In order to understand the shared ontology the user should first look at the different modules (implicitly) defined in the ontology. She can then consult the definitions of the concepts and read the available documentation. The arguments underlying the conceptualization may also be helpful.

As a guideline the user should start to understand the concept hierarchy before looking at the relations between the concepts. Defined axioms and inference rules are likely to be regarded last, because their understandability assumes a satisfactory comprehension level of the concepts, taxonomy and interconceptual relations. The meaning of the instances defined in the shared ontology may serve as good examples.

The analysis of the shared ontology is followed by three alternative activities. The user might organize her local knowledge according to the ontology, modify it in order to improve its usability in the local context or use it to retrieve knowledge from the system. Hence, the underlying ontology can actually change depending whether the user has modified it in any of the parallel activities.

Ontology utilization This activity summarizes the general utilization of the ontology as a means to query and gather local and remote information. Depending on the restrictions imposed by privacy laws, the utilization can be automatically monitored in order to detect usage patterns. Frequency of use of certain entities in the ontology can enable the board to enhance the quality of the conceptual model. In case users miss concepts when formulating their information needs new requirements for the ontology emerge.

The activity continues during the entire local adaptation step and is not proceeded by other activities.

Ontology instantiation Ontology instantiation also known as ontology population is the activity of classifying the available knowledge in terms of the ontology. The local information is one source of available knowledge[7] Other sources are the participants in the process and external providers. With this activity all participants contribute to the collective knowledge available in the system. Users also integrate knowledge they have retrieved from third parties into their own knowledge base. Relations between own and other conceptualizations beyond the already shared ontology might be detected during the integration of external knowledge to the local repository. This activity includes also the provision of mappings between equivalent conceptualizations from different users.

[7] In our case studies the local information consisted of documents, contacts, emails, Web bookmarks etc.

The local and remote information may contain knowledge which cannot be formalized in the shared ontology and thus can lead to the specification of new local requirements for the ontology. Depending on the ontology and the objective of system, information can be organized thematically. In our case studies the ontology comprised a topic hierarchy. Which knowledge is finally encoded in the ontology depends on the design of the ontology and the user needs. We have observed also different instantiation styles. Some users are very enthusiastic and instantiate the ontology very quickly other are more reluctant and shy away from the initial extra effort.

The activity continues during the entire local adaptation step and is not proceeded by other activities.

Specification of new requirements During the local adaptation phase, the more important activities from an ontology engineering perspective occur when the user realizes that the shared ontology does not conform to the requirements for the local application setting and should thus be modified to met these additional requirements. The objective of this activity is to identify and specify the requirements for the local ontology.

The task(s) the ontology is involved in play(s) an important role in identifying the requirements for the local ontology. The possibilities are wide-ranging. As already mentioned in the build phase trained ontology engineers can use ORSD documents to capture the requirements for the ontology in this phase [Sur03]. Other methods to identify requirements have also been mentioned in conjunction with traditional ontology building efforts. [8] Requirements for the ontology can be derived from competency questions as it was suggested in [GF95a]. Existing ontologies might be the driver for new requirements as well. In our case study the identification of requirements for the local ontology by the analysis of existing folder structures was particularly useful. It is important to note that requirements for the local ontology can also come from other users – one participant in the process captures the requirements for the shared ontology of other users as a representative.

The user must decide whether to implement the requirements for the local ontology consulting other users, adapting the local ontology on her own or to submit the requirements to a representative.

Local analysis of additional (local) ontologies Depending on the ontology development scenario the user might have access to other parties' ontologies and integrate their local adaptations in her own application ontology as a resource saving alternative to a new build. This activity is related to the activities defined in ontology building by reuse. As described above in the reuse setting ontology engineers must thoroughly examine the

[8] The users in our case study were mainly missing topics to be able to classify their documents in a enough fine granular manner. In this case the requirements analysis was less structured and mainly driven by the users experiences in instantiating the ontology.

candidate ontologies. The ontology engineer should for example consider which conceptualizations must be change, removed, relocated, which definitions and documentations must be changed etc. Note that the translation of the ontologies involved in this reuse attempt is not an issue in DILIGENT, which assumes the usage of a unique representation language for the shared and local ontologies in the distributed scenario.

Reusing existing ontologies is a feasible alternative for both technically versed ontology engineers, which follow established reuse methodologies to fulfill this task, and eventually less experienced ontology users. The users should analyze the external ontologies in a straightforward manner. The shared ontology allows for examining only certain parts of the ontology and consider only them for reuse. Furthermore, users reuse other ontologies although they might not exactly fit their requirements for the ontology, since they do not agree building it on their own. If only small modifications of the shared ontology are needed to meet the user requirements for the local ontology then users will not consider remote local ontologies.

In any case the result of the activity is the decision from whom to reuse which parts of the remote local ontologies.

Customization of relevant local ontologies The goal and the content of this activity depend on the result of evaluating external ontologies w.r.t. their local suitability. In the extreme fusion case the reused ontology serves only as an input to be completely reorganized. In the case of integration to the shared ontology parts of the remote ontologies can be reused unchanged. Depending on the reuse level the remote ontologies might be subject of more or less radical customization measures. However, a lower customization effort will help the board to find reuse patterns.

The result of this activity is an ontology which can be integrated with the local shared ontology.

Integration of reused local ontologies to the shared ontology Finally the reused remote ontologies should be locally aligned to the shared ontology. This may result in the modification in their shared ontology. Again generalization or refinements can be necessary in order to integrate the reused ontologies.

Additionally to the traditional reuse activity the integration with the shared ontology involves a mapping task. The users keep a direct reference to the reused conceptualizations, with the beneficial consequences that they can access the remote users data and that the board can recognize communalities between different users, and utilize this knowledge when changing the shared ontology.

The result of this activity is an locally adapted shared ontology with adaptations based on remote users ontologies. The origin of the adaptations is stored in mappings between the reused and local ontologies.

Modification of shared ontology The modification of the shared ontology is another option to adapt it to the local requirements for the ontology. Modifications range from small changes *e.g.* adding a new concept or relation to a complete restructuring of the shared ontology. The local requirements for the local ontology can also implicate that an entire new module should be integrated into the shared ontology.

The user should decide which parts of the ontology should be changed and how to implement the desired changes. The changes the user introduces may point to missing abstractions in the existing model.

Furthermore this activity starts from similar assumptions as they were described in the Sensus methodology [SPKR96]. The shared ontology represents far more knowledge than the user actually needs. This implies that the user will remove a number of conceptualizations from the shared ontology and maintain only a smaller part. In this case the board has the challenging task to decide which parts of the shared ontology should be removed because they are not needed and which parts are e.g. to general to be used but helpful for inter-user communication.

This activity includes the conceptualization as well as the implementation of the required changes.

As a result of this activity the local ontology mets the local requirements on it.

Argument provision As far as possible the user should track the reasons why certain modeling decisions were performed in a certain way. We propose the provision of arguments according to a specific model [TPSS05, PSTS04] to capture these decisions. The model defines the process of providing arguments and several kinds of arguments, and aids decision making. While the latter is particularly relevant to collaborative ontology engineering, the first two aspects will help the board to understand the users decisions.

Arguments can range from simple usage examples (e.g., some document could not be classified using the ontology, some query could not be answered by the ontology to a satisfactory extent) to twisted argumentations trading-off the pro's and con's of a decision. The more expressive the argumentation is, the easier it will be for the board to understand the reasons for the decisions and to integrate the newly submitted change requests to the shared ontology. Additionally, users intending to reuse the conceptualization – as aforementioned in the previous, reuse-oriented activities – are provided considerable support to comprehend and use the corresponding ontology correctly.

Evaluation of new local ontology The evaluation procedure is divided into three categories. The syntactic, semantic and pragmatic evaluation of the new local ontology. As the user utilizes the ontology mainly to organize her own knowledge the pragmatic evaluation is predominant in this activity. She can quickly realize whether the proposed way of organizing her knowledge in the shared ontology is sufficient to capture her local knowledge. The user requirements for the local ontology change with time, hence a sufficient

local ontology can become insufficient after some time. In this case the user will start the process again and capture the emerging requirements for the ontology.

Documentation As far as possible the user should document the changes introduced into the shared ontology. Documentation includes the meta data provision like, when a change was performed, who has performed the change, if the change was done on request from an other user etc. Furthermore, brief description of the added conceptualizations will facilitate the boards task.

4.4.2.6 Use Case Specific Actions in the Generic Description

In the use case specific version of the DILIGENT process model six actions are defined for the Local Adaptation stage. The scope of *Understand shared ontology* is broadened by adding evaluation aspects and is now called `Local analysis of shared ontology` emphasizing the different location between creation and usage of the ontology. The activities *Identify communalities* and *Identify missing* are merged and found in `Specification of new requirements`. The activity *Map equivalent* is part of the `Ontology instantiation` and the `Integration of reused local ontologies to the shared ontology`. *Organize local knowledge* is included in the `Ontology instantiation` activity. The use of the ontology is not an explicit activity in the use case version of the methodology, but since it is a major source for the detection of new requirements for the local ontology we introduced it as `Ontology utilization`.

The local adaptation of the shared ontology, is for the user one action, *viz. Change locally*, and spread along four activities `Local analysis of additional ontologies`, `Customization of relevant ontologies`, `Integration of reused local ontologies to the shared ontology` and `Modification of shared ontology` to account for the different ways of realizing it. In case ontologies from other users are reused we use the terminology established in the literature for these purposes. Note that this is a different activity than the modification of the shared ontology, since it implies for instance the comprehension and evaluation of ontologies developed at different sites in the local context.

In compliance with established ontology engineering methodologies we further introduced the activities `Argument provision`, `Evaluation of new local ontology` and `Documentation`, which are not explicit in the use case specific version.

4.4.3 Analysis

As described in the methodology, the board will come together in fixed time lines or when a certain threshold of change requests has been reached. The frequency of this analysis is determined based on the frequency and volume of changes to the local ontologies. They will subsequently analyze the activities which have taken place. They will gather the ontologies from all participating peers on one central peer. The main task of the board is to incorporate the change requests into the core ontology and to identify common usage patterns. Our tool supports the board members in different ways to fulfill their task.

4.4.3.1 Roles

In the analysis stage we can distinguish three roles played by board members: (i) The domain expert decides which changes to the common ontology are relevant for the domain and which are relevant for smaller communities only. (ii) Representatives of the users explain different requirements for the shared ontology from the usability perspective. At this stage, work is conducted at a conceptual level. (iii) The ontology engineers analyze the proposed changes from a knowledge representation point of view foreseeing whether the requested changes could later be formalized and implemented.[9]

4.4.3.2 Decisions

The board must decide which changes to introduce into the new shared ontology at the conceptual level. Metrics to support this decision are (i) the number of users who introduced a change in proportion to all users who made changes. (ii) The number of queries including certain concepts. (iii) The number of concepts adapted by the users from previous rounds.

4.4.3.3 Input

The analysis stage takes as input the ontology changes proposed and/or made by the participating actors. To be able to understand the change requests, users should provide their reasons for each request. Both manual and automated methods can be used in the previous stages. Besides of arguments by ontology stakeholders, one may here consider rationales generated by automated methods, e.g. ontology learning. The arguments underlying the proposed changes constitute important input for the board to achieve a well balanced decision about which changes to adopt.

[9] In the revision stage.

4.4.3.4 Output

The result is a list of the major changes to be introduced that were agreed by the board. Hence, all changes which should not be introduced into the shared ontology are filtered. In this stage it is not required to decide the final modelling of the shared ontology.

4.4.3.5 Activities

The board meets regularly in order to include emerging requirements for the shared ontology to the existing one. To achieve the desired output the board takes different activities namely *Information collection from users Analysis of obtained information Control of previously shared ontology Specification of new requirements* .

We now detail each one of the proposed activities:

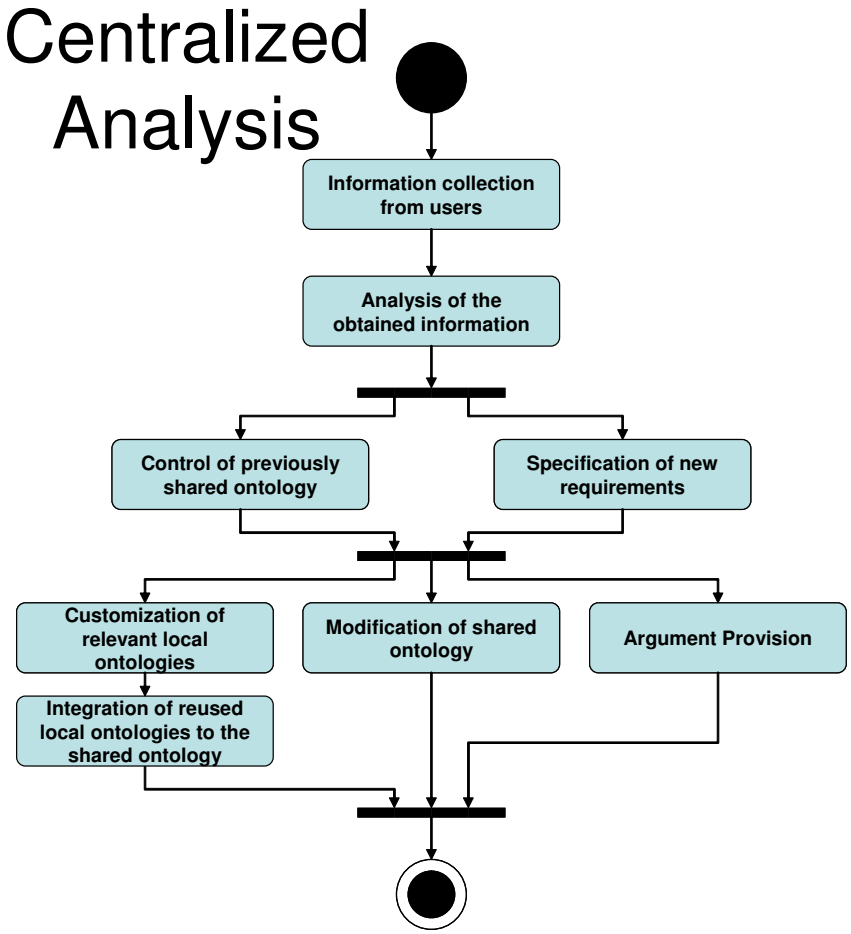

Figure 4.5: Analysis: Activity Diagram

Information collection from users Before the board can identify new requirements for the shared ontology it collects the local ontologies from all participants, the respective argumentation, change requests provided by other means, usage information and finally mapping information.

Depending on the deployed application the gathering of the locally updated ontologies can be more or less difficult. It is important that the board has access to the local changes from users to be able to analyze them.

This activity reminds of the classical reuse approach to ontology building in which candidate ontologies must be gathered. In contrast to the classical reuse approach to build ontologies the ontologies which must be integrated into the shared ontology is given. Furthermore the domain and application scenario are already defined. Usage information for the ontology is available, hence the relevance for the shared ontology is easier to determine. No translation must be performed in order to integrate the ontologies.

Control of previously shared ontology The goal of this activity is to examine the changes introduced in the last cycle. Specifically the board checks how many users have integrated the proposed changes and the tasks the shared ontology was used for. The board can detect if the users accept the common conceptualizations, if the analysis methods are appropriate and if the users understand and agree with the view of the board.

The averaged adaptation rate of concepts from the core ontology and also of concepts from different users is an indicator of how well a concept fits the user needs (*cf.* 4.1). If a concept of the core ontology was not accepted by the users it probably has to be changed. Alternatively, a concept introduced by a user which has been reused by many other users can easily be integrated into the core ontology.

$$\text{adaptation rate} := \frac{\text{No of participant who have locally included the concept}}{\text{No of participants}} \tag{4.1}$$

Specification of new requirements New requirements for the shared ontology can be obtained by analyzing the change requests, the changes in the local ontologies and the arguments provided by the users.

Analysis of obtained information It might also be interesting not only to analyze the final user ontology, but also its evolution. However, with an increasing number of participants this in-depth analysis might be unfeasible. Since analysis takes place at the conceptual level, reverse engineering is usually an important technique to get the conceptual model from the formalized model [GPFLC03].

The number of change requests may be huge and also contradictory. First the board must identify the different areas in which changes took place. Within analysis the board

should bear in mind that changes of concepts should be analyzed before changes of relations and these before changes of axioms. Good indicators for changes relevant to the users are (i) overlapping changes and (ii) their frequency. Furthermore, the board should analyze (iii) the queries made to the ontology. This should help to find out which parts of the ontology are frequently used. Since actors instantiate the ontology locally, (iv) the number of instances for the different proposed changes can also be used to determine the relevance of certain adaptations.

Argument provision As described above we have conceived an argumentation framework to support the discussion taking place in collaborative ontology engineering. The board is also supported in making decisions.

4.4.3.6 Use Case Specific Actions in the Generic Description

In the use case specific version of the process we define six activities for the centralized analysis. The activity *Gather updated ontologies* is broader in the general case and includes the collection of all relevant information from the users, such as arguments and informal change requests. It is called `Information collection from users`. The activity *Analyze changes conceptually* is thus broader in the general case and is called `Analysis of obtained information` accounting for the different ways change requests can be made. We further introduce a controlling activity, `Control of previously shared ontology`, in order to establish a feedback loop in our process.

The activity *Decide on changes to be made* is described as `Specification of new requirements`.

4.4.4 Revision

While we could evaluate the local update and analysis step of our process model already in small experiments the revision phase and local update phase are not well tested yet. Hence, due to the early stage of the project the revision phase and the local update phase are not yet very well elaborated. We here just sketch some general observations and point our future directions of research.

4.4.4.1 Roles

The ontology engineer judges the changes from an ontological perspective, more exactly at a formalization level. Some changes may be relevant for the common ontology, but may not be correctly formulated by the users. The domain experts should judge and

decide whether new concepts/relations should be introduced into the common ontology even so they were not requested by the users.

4.4.4.2 Input

The input for the revision phase is a list of changes at a conceptual level which should be included into the ontology.

4.4.4.3 Decisions

The main decisions in the revision phase are formal ones. All intended changes identified during the analysis phase should be included into the common ontology. In the revision phase the ontology engineer decides how the requested changes should be formalized. Evaluation of the decisions is performed by comparing the changes on conceptual level with the final formal decisions. The differences between the original formalization by the users and the final formalization in the shared ontology should be minimal.

4.4.4.4 Output

The revision phase ends when all changes are formalized and well documented in the common ontology.

4.4.4.5 Activities

Customization of relevant local ontologies Integration of reused local ontologies to the shared ontology Modification of shared ontology Argument provision Argumentation aggregation Evaluation of new shared ontology Documentation and *Distribution of new shared ontology*

Customization of relevant local ontologies After analyzing the changes and assigning them according to the concrete ontology modules they address, the board has to identify the most relevant changes. Based on the provided arguments the board must decide which changes should be introduced. Depending on the quality of the arguments the board itself might argue about different changes. For instance, the board may decide to introduce a new concept that better abstracts several specific concepts introduced by users, and connect it to the several specific ones. Therefore, the final decisions entail some form of evaluation from a domain and a usage point of view.

Integration of reused local ontologies to the shared ontology The customized reused local ontologies must be integrated with shared ontology. Here again it might be necessary

to include abstractions or refinements into the shared ontology in order to be able to integrate the reused ontologies adequately.

Modification of shared ontology Similar to established methodologies the requested changes must be formalized with respect to the expressivity of the ontology. We will not go into detail with this step since it is already described in methodologies referred to in the related work section.

Evaluation of new shared ontology The board will evaluate the shared ontology from an syntactic and semantic perspective.

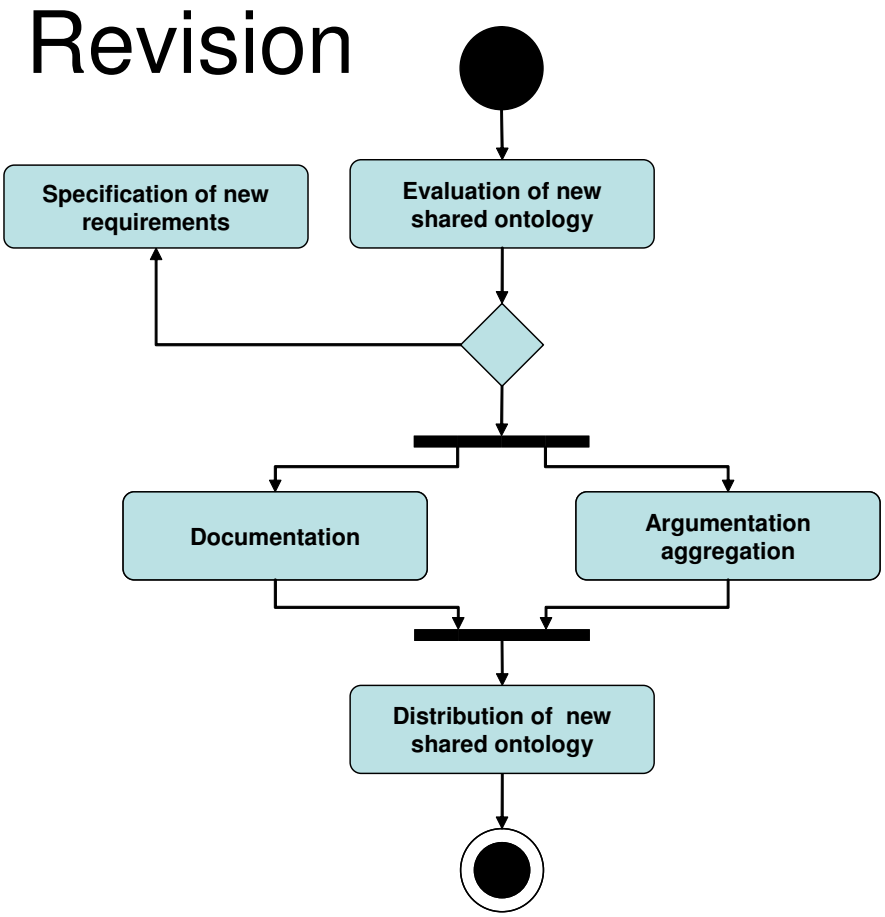

Figure 4.6: Revision: Activity Diagram

Argumentation aggregation As arguments play a major role in the decision process we expect that the changes which are eventually included into the common ontology are supported by many arguments. One of the reasons for keeping track of the arguments is to enable users to better understand why certain decisions have been made with respect to the ontology. Hence, the user should be able to retrieve the most convincing arguments made to introduce a certain change. Here the board aggregates the arguments exchanged during their discussion and makes them more accessible.

Documentation Additionally to the regular documentation suggested for all ontology development process, like authors, number of concepts, purpose of the ontology, we suggest to include the names of the board members, the number of users participating in the ontology evolution, the version very important, the date of revision, in case of conflicting design decisions the voting of the board, the alternatives, the next planned revision date, the voting mechanism, decision procedure and included ontologies.

With the help of the arguments, the introduced changes are already well documented. However, we assume that some arguments might only be understandable for the domain expert and not for the users. Hence, we expect that the changes should be document to a certain level. In particular it should be documented who supported certain conceptualizations and the alternatives.

Distribution of new shared ontology Analogously to step 4.4.3 the shared ontology must be distributed to the different participants. Depending on the overall system architecture different methods can be applied here.

4.4.4.6 Use case Specific Actions in the Generic Description

Existing methodologies describe the activity *Formalization of relevant changes* as four separate activities: `Customization of relevant local ontologies`, `Integration of reused local ontologies`, `Integration of reused local ontologies to the shared ontology` and `Modification of shared ontology`. On the one hand the board will reuse the local ontologies from the users to extend the shared ontology on the other hand user can request changes informally which then lead to modifications of the shared ontology.

We further introduce explicitly the activity for `Argument provision` and for the `Evaluation of new shared ontology`.

The activities *Aggregation of arguments* and *Documentation* are unchanged, though they are renamed to `Argumentation aggregation` and `Documentation`, respectively.

In order to separate activities performed centrally or locally the last activity of this

stage is the `Distribution of new shared ontology` which was previously an activity of the Local Update stage.

4.4.5 Local Update

As a result of the revision stage the participants in the process are aware of the new version of the shared ontology. They must now decide which parts - if any - they use from the new shared ontology. Switching from one ontology to an update incurs effort for understanding the new parts and partly reorganizing the local knowledge base. The gains of updating are lower communication effort and actual information. The incentives for the user to update are higher the more change requests to the shared conceptualizations are included in the shared ontology. Thus the user controls how many of the own proposals are included in the new version and in which way they are implemented. Furthermore the user analyzes all changes to the shared ontology and decides whether to finally integrate the new version with her local ontology.

4.4.5.1 Roles

The local update phase involves only the users. They perform different activities to include the new common ontology into their local system before they start a new round of local adaptation.

4.4.5.2 Input

The formalized ontology including the most relevant change request is the input for this step. We also require as an input the documentation of the changes. For a better understanding the user can request a delta to the original version.

4.4.5.3 Output

The output of the local update phase is an updated local ontology which includes all changes made to the common ontology. However, we do not require the users to perform all changes proposed by the board. The output is not mandatory, since the actors could change the new ontology back to the old one in the local adaptation stage.

4.4.5.4 Decisions

The user must decide which changes he will introduce locally. This depends on the differences between the own and the new shared conceptualization. The user does not need to update her entire ontology. This stage interferes a lot with the next local adaptation stage.

4.4.5.5 Activities

This stage can be divided into the three activities *Control of new shared ontology Local analysis of changes in the new shared ontology* and *Integration of new and old version*.

Control of new shared ontology Likewise the board controlling the acceptance of the shared ontology the user controls the implementation of her own proposals. The user controls whether the proposed changes are implemented in the new shared ontology at

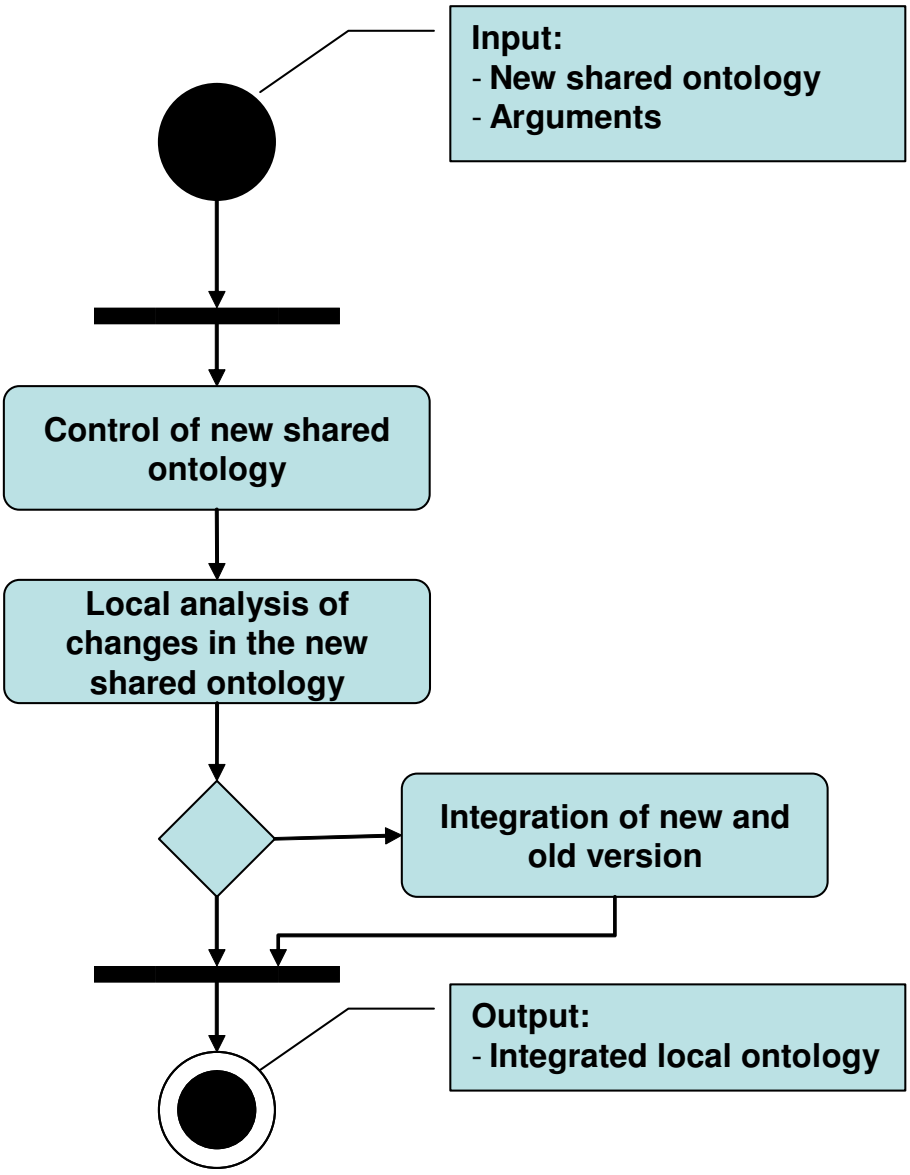

Figure 4.7: Local Update: Activity Diagram

all, conceptually or as proposed. This allows the user to judge which of her proposal are interesting for the community. Furthermore she learns how the board translates the proposals into conceptualizations in the shared ontology.

$$overlapMeasure = \frac{\#integratedChanges}{\#totalLocalRequests} \tag{4.2}$$

$$directOverlapMeasure = \frac{\#directlyIntegratedChanges}{\#totalLocalChanges} \tag{4.3}$$

$$conceptualOverlapMeasure = \frac{\#conceptuallyIntegratedChanges}{\#totalLocalRequests} \tag{4.4}$$

$$\begin{aligned} \#integratedChanges = {} & \#directlyIntegratedChanges + \\ & \#conceptuallyIntegratedChanges + \\ & \#integratedChangesArguments \end{aligned} \tag{4.5}$$

Local analysis of changes in the new shared ontology The user changes locally to the new shared ontology only if her benefits predominate the effort of updating. The analysis of the introduced changes inform her whether the changes effect her or not.

Technically this step requires the construction of a delta view on the ontology.

Integration of new and old version The result of the analysis is the decision to integrate completely or partially the new shared ontology with the existing local ontology. The new shared ontology may contain refinements of existing model. In this case the user should consider to adapt her instantiations with respect to the refinements. The outcome of the controlling activity allows the user to decide which restructuring she must perform in order to stay in line with the new model. The new version can also be a model for knowledge which was previously not covered by the shared ontology. In this case the existing local knowledge can be the source for the population of the ontology in the next stage.

From a technical point of view we could identify several requirements for the acceptance of the process model from the case study. Acceptance and usability of the process model largely depends on the ease of translation from old to new versions. As in other systems the possibility to switch automatically between the different versions of the ontology enhances user experience. The system must support the user to easily integrate the new version into her local system. It must be guaranteed that all annotations made for the old version of the ontology are available in the new version.

The user should be enabled to use from now on the shared model instead of her own identical model. Furthermore, the board might have included a change based on arguments the user was bringing forward, but has drawn different conclusions. Here the user can decide whether he prefers the shared interpretation. Other option might emerge in the course of the case studies.

To ensure user satisfaction, the system must enable the user to return to her old version of the ontology at any time. The user might realize that the new updated version of the common ontology does not represent her needs anymore and thus want to leave the update cycle out. To reach a better acceptance this must be possible and is foreseen in the methodology. The user can always balance between the advantages of using a shared ontology or using her own conceptual model.

After the *local update* took place the iteration continues with *local adaptation*. During the next *analysis* step the board will review which changes were actually accepted by the users.

4.4.5.6 Use Case Specific Actions in the Generic Description

The activity *Distribution of the new ontology* is part of the previous stage in the general version. The activities *Tagging of the old ontology* and *Local inclusion of the update* in the old version of the process model are technically motivated. Though technically necessary, methodologically they are rather part of the `Integration of new and old version` activity as define in the general version. This covers also the *Alignment of old and new versions*. Before the actual integration of the new version with the old version can be processed we introduced a `Control new shared ontology` activity to allow for feedback to the users. Furthermore we left the decision whether to update to the new version explicitly to the user. The `Local analysis of changes in the new shared ontology` is the basis for this decision.

4.5 Requirements for DILIGENT Tool Support

The efficient application of a process model depends on the tool support for its phases. Therefore we analyze each phase, taking into account the defined activities, in order to derive the requirements to support them. Our case study experiences in ontology engineering, literature analysis and our scenario are the main drivers of our requirements specification, therefore in other application scenarios new requirements for tool support can emerge. The requirements derived from activities described in other methodologies are summarized briefly and we focus on those specific to our process model.

Grouped by the process stages we describe for each activity the process specific requirements for tool support. In case an activity duplicates the requirements for tool support of another activity we reference the ones mentioned earlier. As the requirements are later mapped to the functionality provided by the tools we list them and highlight their names, e.g **requirement**.

4.5.1 Build

As mentioned earlier the building stage is based on established methodologies. Accordingly many requirements for the different activities have already been identified. We summarize them in the following and go into detail only for the requirements arising from the *Argument provision* activity.

Available methodologies propose different ways to perform the domain analysis (**Domain analysis**) , *e.g.* domain expert interviews, literature study or workshops. Results of these activities are *e.g.* transcripts, lists of documents, lists of competency questions or mind maps depending on the elicitation procedure. A tool should be able to integrate all information which lead to the requirement specification for the ontology. Additionally the tool should support requirements specification according to the ontology requirements specification document (ORSD) and should store information such as the domain description, the defined sub domains, the reusable ontologies, the competency questions and other relevant meta information.

Ones the requirements are specified the board builds the shared ontology. A tool should support adding and removing of concepts, ontology management tasks, *e.g.* versioning, storing and evolution and ontology visualization tasks (**Ontology editing**) . In case ontologies are reused their integration should be supported, thus import of ontologies serialized in different formalization languages, copy and paste etc..

The board should evaluate the shared ontology according to predefined metrics (**Ontology evaluation**) . On the syntactic level the tool should ensure the correctness of the ontology w.r.t. the chosen representation language. Several metrics have been defined to facilitate ontology evaluation on the semantic level, which should be supported by the tool. The usability of the ontology is tested on the pragmatic level. Integrated feedback mechanisms can help to capture user satisfaction with the ontology.

Argument provision is an activity so far not defined in other methodologies. Therefore we elaborate on the requirements for tool support with more detail. Based on our experiments and a review of requirements for tools supporting arbitrary argumentation [CB88] we identified the following requirements for argumentation tool support. These requirements are later on refer to as (**Argument provision**) .

1. (**Communication**) In our scenario participants in the ontology engineering discussion are distributed. Thus discussion can take place online. This means that all participants should be able to obtain up-to-date information about the exchanged arguments, the agreed upon issues, the issues currently under discussion and the available participants. On the other hand the engineering process can be organized asynchronously. In this case the discussants provide their arguments independently of the others. In this case the tool should highlight the new information and inform participants if someone has opposed or agreed to their suggestions.

2. (**Issue / Idea stack**) The issues and ideas should be grouped according to their

priority. Not all issues/ideas can be discussed at the same time. The tool should visualize which issues have already been agreed upon, which are under discussion and which have been postponed. The status of an issue/idea should be easily changeable.

3. (**Classifiable arguments**) As we have predefined the types of arguments the tool should support the user in selecting them. This can be achieved by templates, which suggest a way of formulating a specific argument. Another option is the automatic classification of the argument types, if the arguments are provided as free text.

4. (**Voting mechanism**) If an issue or idea cannot be agreed upon unanimously, the participants must vote for their favorite. The tool should support different decision mechanisms such as majority voting. The participants should be given a time frame to vote.

5. (**Integration into Ontology Editor**) The integration of the argumentation with an ontology editor is important, as the argumentation is a support activity. The discussants should not be forced to change tools for discussing the ontology and building it. Although argument capturing has immediate and long term benefits, it requires additional effort from the users. Usability aspects are therefore in particular important. It has the further advantage, that reasoning capabilities and different visualization techniques are already available.

6. (**Concurrent ontology visualization**) During the discussion the ontology evolves in different ways. Hence different models of the same ontology exit, while some parts are already agreed by all participants, other are still under discussion. To enable the participants to evaluate and compare the different proposals, the tool should be able to visualize the concurrent versions of the ontology. This includes selection mechanisms, which allow the participants, to select parts of the ontology proposed by one actor, or to exclude parts of the ontology proposed by another actor. Furthermore the agreed parts should be selectable and it should be possible to export these parts in a formal language.

7. (**Moderator**) In online discussion the tool should allow the moderator to decide who talks how long and which issue is under discussion. Furthermore, the tool should provide the moderator with some predefined questions to focus the participants or restrict their contributions.

As elaborated in other methodologies a tool supporting (**Documentation**) should facilitate the description of the ontology on the general level as well as on the concept level. The tool should ensure the accessability of the documentation. As documentation is particular important for reuse scenarios of the ontology, the documentation should come in different languages and include links to the argumentation underlying the modeling decisions.

4.5.2 Local Adaptation

In order to select the requirements for tools motivated by the activities in the local adaptation step we distinguish two main types of users. The less frequent type is the user with ontology engineering competence who analyzes her personal needs, conceptualizes and formalizes them. He uses established ontological guidelines [GW02] in order to maintain soundness and validity. Besides, she annotates her knowledge according to her locally extended ontology. Accordingly, the respective requirements are very similar to the ones already defined in the previous section.

The more common type of user is a non-ontology engineer. The shared ontology is regarded as a predefined categorization schema, such as the categorization he defines in her daily work (*e.g.* her folder structures). Annotations of documents are primarily provided in the form of assignments to a specific category. A good shared ontology is thus one, which allows them to structure their documents and other information.

The analysis of the shared ontology can be facilitated by an appropriate visualization (**Visualization**) . A detailed requirements analysis for ontology visualization is beyond the scope of this thesis. However, in particular for our scenario it is important that the user has access to the arguments supporting certain parts of the ontology. As the shared ontology grows in size highlighting of concepts most relevant for the user context becomes necessary.
In some application scenarios the user might not be aware that the information is represented in an ontology. Therefore it depends predominantly on the specific use case for the application, which kind of user interface is used to efficiently use the ontology (**Ontology use**) . Important aspects are a domain specific query interface and answer visualization. Detailed requirements for tool support should be defined application dependent.

Some users organize their local information according to the ontology. One aspect is that the local information must be accessed (**Access to local information**) . The general solution to link a source with an repository is the introduction of mediators [AvE04, Wie92], which extract information from the sources and transform it in the target representation language. In order to support later phases in our process we require that provenance information is stored. Another aspect regards the annotation of the local information according to the ontology (**Annotation**) . As the manual instantiation of an ontology requires sparse human resources an automated support is desirable. A prominent information source in the context of the semantic web are web sites. For this particular case [Han05] has identified the requirements for annotating web sites.

Local information is not only a source for ontology population but can also indicate that the ontology itself should be changed. The emerging requirements for the ontology are captured following the argumentation model (**Argument provision**) . For example in our case studies the users organized their information in classification structures according to their individual views. Requirements for the shared ontology arise mainly from a mismatch between the locally available classification and the shared ontology. The new

requirements for the local ontology are met by locally editing the shared ontology or by consulting and integrating ontologies from other users (**Ontology editing**) . Automatic generation of an ontology based on locally available information can facilitate the former strategy (**Ontology learning**) . A prerequisite to follow the later strategy is the availability and accessibility of remote ontologies (**Access to remote ontologies**) . Although in our case studies all participants represented their ontologies in the same representation language translation support is required in other cases. The assessment (**Visualization**) , including documentation analysis, argument analysis, concept analysis, of the retrieved ontologies is followed by their local integration, including entity selection, entity mapping, entity alignment (**Integration**) , and provenance storing (**Provenance**) . The evaluation of the adapted ontology is performed according to the predefined measures (**Evaluation**) and documented to facilitate future usage (**Documentation**) .

4.5.3 Analyzing

The analysis of the user requests is preceded by the collection of information (**Access to remote ontologies**) . The board might trigger the information collection more frequently than having joint meetings, as they gather only if a certain number of changes has been introduced. Favorably the local ontologies can be obtained directly from the users. Informal requests of the users should be collected centrally, e.g. a board member is responsible to collect the information. If the local update procedure should be performed semi-automatically it is essential that provenance information is delivered with the local ontologies and the arguments.

In order to analyze the obtained information the board needs to visualize (**Visualization**) and group it according to different measures (**Clustering**) . Experience from our case studies suggest that the analysis can start with a simple alphabetical sorting of the introduced ontology entities. However, more sophisticated groupings according to similar topics or similar aspects of the ontology are desirable. In particular arguments and issues provided as part of the argumentation framework, which point in the same direction should be grouped together. Progress information and responsibilities should be assigned to the change requests, if their number is very large .

Another important aspect is calculation of the defined metrics (**Measurement calculation**) , which depends on the availability of provenance and usage information. The metrics defined w.r.t the utilization and adaptation of the shared ontology support the board judging their previous revision. The metrics defined w.r.t the requested changes allow the board to identify the parts of the shared ontology which need a revision. Based on the change requests and the local changes to the shared ontology the board is able to specify new requirements for the next version of the shared ontology.

4.5.4 Revision

The conceptualizations following from requirements identified in the previous phase must be implemented in the revision phase. This requires basic editing functionality (**Ontology editing**) from a supporting tool. The ontology changes must be resolved taking into account that the consistency of the underlying ontology and all dependent artifacts are preserved and may be supervised [SMMS02].

The integration of the new shared ontology with the user's local ontologies should be as easy as possible (**Provenance**) . The changes to the shared ontology based on direct integration of user proposals should reference the original change. This is also required for changes which are inspired by user changes or arguments. In this case the provenance information should include all arguments and changes which lead to the specific decision. Similarly the trace between summarizing arguments and summarized arguments should be maintained.

The final changed shared ontology is made available to the users (**Publishing**) . Either it can be published centrally and the users are required to obtain on their own request or it is send to to the user directly.

4.5.5 Local update

The users of the shared ontology have already obtained the shared ontology and decide in the local update stage if they update to the new version or not. In order to analysis the new version of the shared ontology they need different (**Visualization**) options. In particular the parts of the ontology which have changed in comparison to the users actual version should be highlighted. The changes based on the user's own requests are of particular interest to her. The user's requests can be integrated into the shared ontology, either directly, conceptually or based on arguments. The evaluation of the control measurements can help the user making her decision (**Measurement calculation**) . The identification of the directly integrated ontology entities is straightforward and the *directOverlapMeasure* can be calculated. In order to calculate the *conceptualOverlapMeasure*, the local changes which are not directly integrated must be identified and possibly (automatically) mapped onto changes introduced in the new shared ontology. The changes introduced based on user arguments can be recognized if they are represented in the argumentation ontology.

The user finally decides which changes to the shared ontology should be integrated locally (**Local integration**) . For this activity it is important, that the consistency of the local model is kept. The instantiation according to the new local model should not result in a loss of information in comparison to the previous local model. However, if the user realizes that h decision to update was a mistake a possibility to restore the old status is required.

Summarizing, we have identified the following requirements on tool support:

- Access to local information
- Access to remote ontologies
- Argument provision
  - Classifiable arguments
  - Communication
  - Concurrent ontology visualization
  - Integration into Ontology Editor
  - Issue / Idea stack
  - Moderator
  - Voting mechanism
- Annotation
- Clustering
- Documentation
- Domain analysis
- Evaluation
- Integration
- Measurement calculation
- Ontology editing
- Ontology evaluation
- Ontology learning
- Ontology use
- Provenance
- Publishing
- Visualization

4.6 Argumentation Framework for DILIGENT

In this section we describe how we specifically investigated whether some argumentation structures dominate the progress in the ontology engineering task and should therefore be accounted for in a fine-grained methodology.

4.6.1 Threads of Arguments

A central issue in the DILIGENT process is keeping track of threads of exchanged arguments. We can identify several stages in which arguments play an essential part:

- Ontology is defined as "a shared specification of a conceptualization" [Gru95]. Although "shared" is an essential feature, it is often neglected. In DILIGENT experts exchange arguments while **build**ing the initial shared ontology in order to reach consensus;
- When users make comments and suggestions to the control board, based on their **local adaptations**, they are requested to provide the arguments supporting them;
- while the control board **analyses** the changes introduced and requested by users, and balances the different possibilities, arguments are exchanged and balanced to decide how the shared ontology should change.

There is evidence that distributed ontology development can be rather time consuming, complex and difficult, in particular getting agreement among domain experts. Therefore, one needs an appropriate framework to assure it in a speedier and easier way. In order to provide better support, one needs to identify which kind of arguments are more relevant and effective to reach consensus. The Rhetorical Structure Theory (RST) can be used to classify the kinds of arguments most often used and identify the most effective ones.

4.6.2 RST Example

The RST has already been introduced in Section 2.5. In the examples provided within the case study section we will highlight the different elements of RST in the following way.

span nucleus ... relation indicator ...
`span satellite` **Relation**

On the one hand we have presentational relations, such as **background** that increases the ability of the reader to comprehend an element in N, **evidence**, where reader's comprehension of S increases his/her belief of N, **justify**, **restatement**, **summary**, etc. On the

other hand we have subject-matter relations, such as **elaboration**, where S presents additional detail about what is presented in N, for instance set::member; abstraction::instance; whole::part, object::attribute, etc., **evaluation**, **purpose**, **solutionhood**, etc. There are also other relations that do not carry a definite selection of one nucleus, such as **contrast**, where the reader recognizes the comparability and differences in situations described in two N, etc.

The analysis process is intended to give a structured, definite way for a person to understand the text to state a part of what that understanding includes. Sometimes one may not find some structural role for every element of the text. A text may have more than one analysis, either because the observer finds ambiguity or finds that a combination of analyses best represents the author's intent. The analysis gives an account of textual coherence that is independent of the lexical and grammatical forms of the text.

The available tools are tables that explain the relations between spans of text. Therefore the analysis process is manual, intensive and requires understanding of natural language.

4.6.3 Analysis in the Biology Domain

In this section we report how the hypothesis underlying DILIGENT argumentation model has been developed. We have found in the field of biology a taxonomy that has been evolving since 1735 for over 200 years, following a DILIGENT 5-step process. A thorough analysis led to a well-defined subset of RST arguments which allows to explain most of the evolution of the biology taxonomy. In the following Section 4.6.4 we describe experiments which have been carried out to provide evidence that our approach to discuss ontologies is applicable in practice and the reduction of arguments leads to a better process instantiation and better results.

Based on the RST analysis of real arguments that are exchanged and used to support changes in this taxonomy, we formulated as hypothesis that there is a subset of arguments that can focus, speed and ease this kind of ontology engineering. In order to prove our hypothesis we performed an *in situ* experiment in two rounds. In the first one participants were not constrained. In the second one participants were requested to use the subset of arguments that had been found more effective in the first round. We show the improvements that were achieved using the restricted set of arguments, proposed in the fine-grained DILIGENT model of ontology engineering by argumentation.

The taxonomy of living things is essential for those studying, classifying and understanding life. When we analyse its evolution since 1735 one notes that it completely follows the 5-step DILIGENT process. It was initially proposed/**built** by Linnaeus based on phenetics (observable features). Each branch of the tree can have at most 26 levels, depending on how rich a taxa is, in terms of number of beings sharing a given classifying feature. Since the initial proposal, the taxonomy has changed a lot. Let us take the "highest" level: kingdom. Initially two taxa were identified: animals and plants. When

microorganisms were discovered the moving ones were classified in the animals kingdom and the colored (non moving) ones in the plants kingdom. A few of them were classified in both kingdoms. Users were **locally adapting** the taxonomy for their own purposes. To more easily identify organisms in both classes, Haeckel (1894) proposed a new kingdom to more easily identify them, the Protista kingdom. This still exists today and is regarded as a "junk-basket" category.

Naming is an important issue. Lineaus binomial system (genus and species) is still in use, because it can univocally identify a given being in the taxonomy.[10] Given the difficulty and similarity of some names, the ever evolving new knowledge about ever growing number of life forms, and the difficulty of making available up-to-date knowledge to all stakeholders about so many life forms, several problems in designing and managing this complex and live/dynamic taxonomy arose. For some time, names of plants and animals have been controlled by different **boards**, that have to some extent, recorded the problems and solutions found for each kingdom. They receive requests for changes, **analyse** them, balance pros and cons, decide upon the most adequate changes to introduce and **revise** the taxonomy accordingly. Once a new version is made available users should use it**/locally update**.

After being divided for two centuries and being controlled by two different boards, there were some communication problems between the two communities. Given the availability of online information about lifeforms and the need to exchange information about new results, the need to develop a common language and a BioCode arouse. This effort is now beginning.

So, the evolution of the taxonomy is driven by a specialized set of users, taxonomists, and the revision is loosely controlled by appropriate boards, that make new versions available for all users.

In this case the central board is the scientific community, the peers, who **analyse** the different proposals to explain new knowledge and accomodate new life forms, and once in a while **revise** the common understanding of the domain.

One can summarize the major force for reorganization of the taxonomy over time as the identification of important classifying features and gathering all beings sharing a given value for that feature into that class. For instance, the classical version by Whittaker (1969) recognizes 5 kingdoms: Monera, Protista, Plantae, Animalia and Fungi. Regarding all eukaryotic organisms, Plantae, Animalia, Fungi and Protista, the first three, classify multicellular organisms according to nourishment, autotrophic, heterotrophic and saprotrophic, respectively. Fungi were promoted from one subclass (taxa) in the Plantae kingdom to a kingdom of its own.

However, there are currently more advanced classifications, that is, several classifications coexist. Therefore, classes can be promoted, moved, folded, deleted, merged, renamed, etc. as more is known about life on earth.

[10] One can reuse names in different kingdoms.

Currently, given the advances in molecular biology, the tendency is to use a cladistic approach to specify the taxonomy of life, in which the taxonomy is organized according to the evolutionary relationships between live forms based on derived similarity. In a cladogram, each split is ideally binary (two-way), and all the organisms contained in any one clade share a unique ancestor for that clade. This entails a major reorganization of the Tree of Life. The reason is that the design decisions are radically different from the previous approach to model the Tree of Life.

EXAMPLE When analysing the arguments exchanged by taxonomists to change the names and organization of the taxonomy one can perceive its vast array and complexity.[11]

> ...*Acinetosporaceae*, including the genera `Acinetospora, Feldmannia, ...` **Elaboration**
> *This group* forms a well-supported clade in molecular trees based on `rbcL data`. **Evidence**
> So far, trees from nuclear ribosomal data do not reveal `them` as *a well-supported group* **Antithesis**
> but are not contradictory to *their* recognition. **Concession**
> ...

DISCUSSION The analysis of the arguments driving the evolution of the taxonomy of life on earth led to the assumption that RST could be useful to analyse arguments exchanged in ontology building process in distributed environments.

From an arguments point of view, the focus of this paper, we can see that although elaborated, there are a few arguments in the biology case study which play a major role, such as examples/evidence, counter examples, elaboration, alternatives and comparisons to convey a certain decision.

4.6.4 Evaluation of Argumentation Framework

In order to substantiate our hypothesis that an appropriate argumentation framework can facilitate the ontology engineering process, we pursued experiments in a computer science department, viz. at the Institute AIFB[12]. Arguments in collaborative, distributed settings take place in a social environment. Therefore organizational issues are non negligible and were also taken into account.

We performed two experiments: in the first, participants were not constrained in any way; in the second, participants were asked to (1) use a subset of arguments, those that that had been found more effective in the first round,(2) and were given stricter rules, and a better environment to conduct their discussions. The task in both sessions was to

[11] Example taken from `http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/index.cgi?chapter=CHANGETOCLASS`

[12] see `http://www.aifb.uni-karlsruhe.de/`

build an ontology, which (1) represents the knowledge available in the research group, (2) can be used for internal knowledge management, (3) and makes the research area comprehensible for outsiders. Both experiments lasted each for one hour and a half. From the eleven participants - all from the computer science department, thus domain experts - three were unexperienced in ontology engineering. Seven of them were very active in both discussions. Concepts were only added after argumentation and some consensus was achieved.

4.6.4.1 First Experiment

The goal of the first experiment was to identify the dominant arguments used to push forward ontology development.

1. **Setting:** The participants met in a virtual chat room. Each one had their own client and all of them could see the current ontology. All arguments were exchanged via the chat room, no other forms of communication were allowed. A moderator was responsible to remind people to stay on the subject and to include the modelling decisions into the formal ontology which was visualized on a web page. At this stage very few procedural and methodological restrictions were a-priori imposed. The subjects were instructed of the high level goal of the experiment, of the procedure and of their goals.

2. **Example:**[13] An excerpt from the real dialogues taking place:

   > ... **sa:** i dont care whether *someone* plays baseball or not when I am modelling `research domain`. **Evaluation cs: sa** just an example... **Circumstance ct:** maybe it is the purpose of *the website*, that people get also `informed about hobbys` **Purpose cs:** so we have person **Restatement jt:** what I find a bit more interesting is `the conference problem` **Motivation** ...

3. **Result:** In the beginning participants brought forward different kinds of arguments, like background knowledge, examples, elaboration and so on. This led to different argumentation threads and the participants were discussing different topics at the same time. At some points there were 4 threads at the same time, most of the time there was more than one, including procedural and noise. Therefore, discussion was very tangled and at some points rather difficult to follow. Topics which were discussed included: the appropriate formalism to model the ontology, detailed elaboration of leaf concepts, which top level concepts to begin with, philosophical modelling decisions (roles vs. multi inheritance), which are the main modules, topic lists etc. From time to time participants called for a vote. However a decision was seldom reached. The moderator interacted only rarely in the discussion,

[13] We have changed the transcripts a bit, for the sake of readability.

because timely moderating multiple threads is very difficult: by the time an intervention was issued two or three other interventions from participants had already been issued. As a result, a core ontology with two concepts, Role and Topic, was agreed upon.

LESSONS LEARNED: We analysed the discussion with the help of RST. Table 4.1 lists the frequency of the different arguments exchanged during the experiment. We could identify the arguments which had most influence on the creation of the ontology, *viz.* **elaboration**, **evaluation/justification**, **examples**, **counter examples**, **alternatives**.

With respect to the experimental setup we identified the following problems: (1) Participants started too many discussion threads and lost the overview, (2) the discussion proceeded too fast, hence not everybody could follow the argumentation, (3) the moderator was too reluctant to intervene, (4) there was no explicit possibility to vote or make decisions. Even in this setting with participants sharing a very similar background knowledge, the creation of a shared conceptualization without any guidance was almost impossible, at least very time consuming. We concluded, that a more controlled approach to discuss ontology design decisions is needed, which structures the process and the moderation.

4.6.4.2 Second Experiment

The goals of the second experiment, were to underline that with an appropriate argumentation framework the ontology creation proceeds faster, more effectively and the resulting ontology represents a shared view.

1. **Setting:** In the second experiment participants were asked to extend the ontology built in the first round. In this phase the formalism to represent the ontology was fixed. The most general concepts were also initially proposed, to avoid philosophical discussions. The initial ontology defined the modelling primitives for topics and the different roles people are involved in. For the second round the arguments **elaboration**, **examples**, **counter examples**, **alternatives**, **evaluation/justification** were allowed.

   The participants in the second case study joined two virtual chat rooms. One was used for providing topics for discussion, hand raising and voting. The other one served to exchange arguments. When the participants - the same as in the first experiment - wanted to discuss a certain topic *e.g.* the introduction of a new concept, they had to introduce it in the first chat room. The topics to discuss were published on a web site, and were processed sequentially. Each topic could then be expatiated with the allowed arguments. Participants could provide arguments only after hand raising and waiting for their turn. The participants decided autonomously when a topic was sufficiently discussed, called for a vote and thus decided how to model a certain aspect of the domain. The evolving ontology was again published on a web site. The moderator had the same tasks as in the first experiment, but was more

Arguments	**First Round**	**Second round**
Elaboration	24	36
Eval. & Just.	14	33
Contrast & Alternative	12	17
Example	12	9
Counter Example	10	8
Background knowledge	9	3
Motivation	5	
Summary	5	3
Solutionhood	4	8
Restatement	3	6
Purpose	3	
Condition	2	
Preparation	1	
Circumstance	1	
Result	1	
Enablement	1	
List	1	1
Concepts agreed on	2	10
Relations agreed on	3	0

Table 4.1: Arguments used and outcome

restrictive. Whenever needed, the moderator called for an example of an argument to enforce the participants to express their wishes clearly.

2. **Example:** An example from the arguments window:

   > . . . **cs:** We have done quite a bit of research in distributed knowledge management (DKM) lately. So I suggest DKM as a topic plus a subtopic "peer to peer" (P2P) **Elaboration ah:** I suggest *knowledge management (KM)* as super concept of `DKM` because `every DKM` is a kind of *KM* **Elaboration, Justification jt:** Well I am now wondering whether *P2P is DKM*, because `File exchange` is not always KM is it? **Counterexample ph:** I suggest *Distributed Comp. (DC)* with `P2P and Grid` as subtopics; *DKM* as subtopic of `DC and KM` **Elaboration do:** PRO **ph** : because his approach *separates KM* and `distributiveness` **Justification, Evaluation cs:** I'd like to agree to **ph** and **do** suggestion. . . . **ad:** km is a kind of kp **Elaboration h:** can you elaborate? **schm4704:** I think we can subsume kdd and km under kp, because both are disjoint, and still both related to knowledge processing (as far as I see it). **Justification, Comparison pc:** knowledge managament is about managing knowledge, structuring, organizing etc. to organize it you need to process it so for sure KP is more special that KM **Elaboration, Justification, Comparison ad:** I did not understand your argument! Why should kp be more specal than km. If I manage something than it is also a kind of process **Counter Example pc:** KP is a part of KM so we should model it as a subpart or something **Justification** . . .

3. **Result:** As expected the discussion was more focused, due to the stricter procedural rules. Agreement was reached quicker. A total of ten new concepts were agreed on. With the stack of topics which were to be discussed (not all due to time constraints), the focus of the group was kept. Some relations were proposed, but they were not agreed upon.

   From a methodological point of view, one can classify the ontology engineering approach followed as **middle-out**. The restricted set of arguments is easy to classify and thus the ontology engineer was able to build the ontology in a straightforward way. It is possible to explain new attendees why a certain concept was introduced and modelled in such a way. It is even possible to state the argumentation line used to justify it. The participants truly shared the conceptualization and did understand it. In particular in conflict situations when opinions diverged the restriction of arguments was helpful. In this way participants could either prove their view, or were convinced.

LESSONS LEARNED: Our experiments provide strong indication – though not yet full-fledged evidence – that a restriction of possible arguments can enhance the ontology engineering effort in a distributed environment. In addition the second experiment underlines the fact that appropriate social management procedures and tool support help to reach consensus in a smoother way. From an RST analysis perspective, the fact that the discussion was more focused eased the task enormously.

Another rather interesting conclusion is the fact that a middle-out approach to ontology building comes naturally for people with knowledge engineering skills when given an appropriate work environment. Moreover, middle-out combined with appropriate argumentation and management can be used to quickly find a shared, consensual ontology even when participants must provide all and only written arguments.

The process could certainly be enhanced with better tool support. Besides the argumentation stack, an alternatives stack would be helpful. Arguments in particular **elaboration**, **evaluation & justification** and **alternatives** were discussed heavily. However, the lack of appropriate evaluation measures made it difficult, at some times, for the contradicting opinions to achieve an agreement. The argumentation should then be focused on the evaluation criteria. The evaluation can take place off-line, or can be based on modelling advices from practical experience. Discussion can proceed. As to the use of the RST to analyse real dialogues, instead of carefully written texts, one should mention, in particular in the first round were the discussion was rather tangled, that it was rather difficult to classify at some parts. However, the restricted set is easy to identify and we conjecture that the provision of template arguments will ease the task further. In both rounds one should stress the lack of tools to automatize it, although one can foresee the difficulty, since this kind of analysis requires deep NL understanding.

4.7 An Argumentation Ontology for DILIGENT Processes

In the previous section we have motivated the relevance of argumentation for the ontology engineering process. The case studies show that exchange of arguments constitutes a major part in collaborative ontology building. We have further shown, that not all types of arguments advance the building process.

This observations from our case studies are inline with the experiences made in software and requirements engineering. There, extensions of the IBIS methodology[SMD02] are used to capture design deliberations, thus make them traceable, and formal models have been developed to allow for structured queries on the arguments[RD92]. They have shown that formal argumentation models enhance traceability of design decision, help in conflict resolution, enhance reusability and facilitates the integration of new participants in the design process. Although, these models are very general, we have identified several requirements – further elaborated in section 4.7.3 – for argumentation support and its

formalization which are unique for OE processes:

1. General argumentation models allow for all types of arguments and are very flexible. However, we have shown that a restricted set of arguments can facilitate OE processes [PSTS04], thus a formal model for OE should take this into account.

2. Within general argumentation models, inconsistencies in the discussion can not be easily detected, since arguers do not formalize their arguments. Ontologies are themselves formal models, thus inconsistencies should be considered during the discussion.

3. Ontology Engineering is often augmented with input from Ontology Learning[MS01]. No methodology provides an integrated view on manual and automatically created ontologies. An ontology learning algorithm can be seen as an agent providing arguments for design decisions. This should be regarded an integral part in a formalized argumentation model for OE.

Threads of arguments A central issue in the DILIGENT process is keeping track of threads of exchanged arguments. We can identify several stages in which arguments play an essential part:

- Ontology is defined as "a shared specification of a conceptualization" [Gru95]. Although "shared" is an essential feature, it is often neglected. In DILIGENT experts exchange arguments while **build**ing the initial shared ontology in order to reach consensus;

- When users make comments and suggestions to the control board, based on their **local adaptations**, they are requested to provide the arguments supporting them;

- while the control board **analyzes** the changes introduced and requested by users, and balances the different possibilities, arguments are exchanged and balanced to decide how the shared ontology should change.

4.7.1 The argumentation process

The DILIGENT argumentation process has been defined has a result of our observations in the case studies. It follows the general recommendations for the organization of group discussions. The process supports asynchronous and synchronous ontology engineering discussions. In asynchronous ontology engineering (OE) discussions the participants do not meet at the same time to discuss, while in synchronous discussions all participants are available at the same time.

4.7.1.1 Roles

A moderator should guide the argumentation process. The moderator ensures that the participants provide relevant argument types, that they stay focused on the discussion, that they adhere to the decision procedures, and that everybody is allowed to argue. These rules apply to asynchronous as well as synchronous OE discussions. If the ontology engineering discussion is synchronous the moderator has the additional task to ensure that one issue is discussed at a the time and that a discussion is followed by a decision.

4.7.1.2 Input

Depending on the stage in which the an argumentation process is initiated, the argumentation prozess takes as input the ontology requirements specification document (ORSD), the shared ontology and the arguments provided locally.

4.7.1.3 Output

The objective of the argumentation process is, that the participants agree on a shared conceptualization for their domain. Besides the shared ontology, the argumentation process results in list of arguments supporting the design decisions.

4.7.1.4 Activities

The argumentation process is divided into the following five activities *Choose moderator*, *Choose decision procedure*, *Specify issues*, *Provide arguments and ideas*, and *Decide on issues and ideas* which are performed sequentially. It is not necessary to specify all issues before the provision of argument, though. New issues can emerge, while others are already formalized.

Choose moderator The participant of the ontology engineering discussion choose a moderator. The basic rules for moderation also apply for ontology engineering discussions. The moderator should not participate in the discussion with contribution to the content, but organizes the discussion. The moderator does not take part in decision, but organizes the decision process.
Each of the participants can take the role of the moderator and the moderator role can move from one participant to the next.

Choose decision procedure The participants should agree on a decision procedure. On the one hand, they should agree on a voting mechanism to decide which option should be chosen. Well know voting mechanisms are the

majority based model All participants have the same voting weight. An issue is accepted if more than e.g., half or two-thirds of the participants accept it.

dictator model One participant decides if an issue is accepted or not

core veto model A core group of participants has a veto, if they do not agree with the majority vote.

On the other hand they should decide, when they want to vote on the available options. They can vote in certain time intervals or if no new arguments have been brought forward for some time. Depending on the scenario the one or the other model can be more adequate.

Specify issues The participants start identifying the issues they want to discuss. The issues define the conceptual requirements for the ontology. All requirements defined in the ORSD should be inserted as issues. The competency questions can serve as a starting point for a list of issues. New issues can emerge during the discussion. Issues which are specification or elaborations of already existing issues, should be classified as elaborations. The issues should be grouped according to the sub-domains they belong to. Any participant should be allowed to bring insert new issues. The decision, wether an issue is relevant and should be modeled should not be discussed during the issue generation phase. The issues should be grouped according to their priority for the application. In particular in synchronous discussions not all issues can be discussed at the same time. If there is no other option the First-in First-out (FIFO) principle can be applied to determine the sequence of the issues.

Provide arguments and ideas The participants discuss the issues, and bring forward their arguments in favor or against an issue. The possible kinds of arguments are specified in the *Argumentation Ontology*. The moderator asks the participants to reformulate their arguments, in case they do not correspond to the allowed argument types. The participants should first agree that an issue is relevant, before they suggest ideas to formalize the issues. Arguments should be concise. The ideas represent the formalization of an issue. If a formalization depends on more than one issue, all relevant issues should be agreed first. Each argument can only be brought forward ones. Participants can agree or disagree with arguments, ideas or issues. They should not repeat them.

In case of conflict the participants should first determine the type of conflict. The different types of conflict have been described in [SG89] and adapted to ontologies according to [ASvE04]. Table 4.2 summarizes the different types.

The solution strategies for the four cases can be:

Consensus Agreement on Idea and Issue level

Correspondence Agreement on Issue but not on Idea level. They must agree on a common formalization.

Conflict Two issues should be model in the same way. They should belong to different modules in the ontology.

Contrast Different Issues refer to different Ideas. The issues should belong to different modules in the ontology.

Concepts	Terminology: same	Terminology: different
same	**Consensus** Experts use terminology and concepts in the same way	**Correspondence** Experts use different terminology for the same concepts
different	**Conflict** Experts use same terminology for different concepts	**Contrast** Experts differ in terminology and concepts

Table 4.2: Types of conflict according to Shaw and Gaines, 1989

Decide on issues and ideas The participants can agree, disagree or postpone the discussion on an issues and an idea. Only idea which are agreed, are part of the shared ontology. Decisions follow the agreed decision procedure.

4.7.2 Use Case

Before we describe the requirements for a formal model to support the argumentation in DILIGENT processes we introduce a number of use cases for the ontology.

4.7.2.1 Traceability

As new partners get involved into the ontology building process, modelling decision are discussed more than once, since the modelling reasons of the existing version are not

Inconsistency	Description
Idea inconsistency	Arguer introduces Idea1 and Idea2 which are inconsistent
Argumentation inconsistency	Arguer argues first in favor and then against an issue. The lines of reasoning followed by the arguer lead to inconsistent ideas
Position inconsistency	Assuming Issue/Argument 1 and 2 are contradicting. An Actor produces a position inconsistency when he votes in favor of Issue/Argument 1 and then introduces Issue/Argument 2

Table 4.3: List of possible inconsistencies

documented. Although, the actors in the current OE discussion present reasons for design modification, the number of e-mails makes it infeasible to retrieve them at a later stage. The ability to present the reasons and arguments for a modelling decision to the **new entrants** could speed up the design process. A similar problem arises, when the ontology is **revised** and the ontology engineers need to recall the reasons for the previous design. The users of the ontology can as well profit from a well documented ontology for a better understanding. Currently, they rely on the sparse explicit documentation, since documentation is a time consuming, often neglected task. A structured integration of the ongoing discussions can ease it.

Another issue is size. The current version of PROTON has more than two hundred concepts. Therefore, it is difficult to track which parts of the ontology are agreed and which are not. In an OE discussion actors often agree only implicitly with a certain modelling decision. For example a participant proposes B as subconcept of A without explicitly agreeing with A.

PROTON had to be build from scratch. Although there are a number of ontologies available on the Semantic Web, the availability of an ontology is not sufficient for it to be reused. Only if the design rationales behind the model are available to others, can ontologies easily be included into new applications.

4.7.2.2 Inconsistency Detection

During the argumentation process different participants exchange their opinions about the issue under discussion. A requirement on an efficient discussion is, that the arguments one participant brings forward are consistent with her previous arguments. A participant may change her opinion, but then he should discard earlier contradicting arguments. A model to conceptualize arguments should be able to detect at least some inconsistencies and point the arguer to the contradicting arguments 4.3.

4.7.2.3 Argument Selection

In the applications for the SEKT case studies the user of PROTON may wonder why some concepts, etc. were introduced in the ontology or he may ask why certain modelling decisions were made. However, even when we trace the underlying arguments, some of them may be very detailed and not understandable to normal users. Hence, if a user asks for the arguments underlying the ontology modelling decisions it would be beneficial to provide an answer which best fits the users needs. In this case we can assume that the best answer to such a query would be one which convinces the requester most. The selection of the appropriate arguments is only possible if not only the argumentation but also the arguments are formalized. Then we can build on models as presented in [Hun04] that show how formal argumentation trees can be pruned to best correspond to the users wishes. On the other hand in tangled discussion it is not always obvious which proposal receives the strongest support. [GK97] presents a formal model to establish the winner of a discussion.

4.7.3 Requirements

We have identified several requirements for our *Argumentation Ontology* from the the SEKT PROTON case study and others where we have been involved such as IEEE SUO. Before we describe the ontology in the next section we now develop its requirements for it.

1. **Use common vocabulary** Research in argumentation and its visualization has a long history and is a mature field (*cf.* 2.5.1.2). To enhance acceptability for the ontology usage of the established vocabulary is essential.

2. **Focus on relevant arguments** As observed in section 4.6 the restriction of available argument types can focus and speed up OE discussions. Hence the ontology should not model all possible kinds of arguments of a discussion, but focus on the relevant ones. This view is supported by [BGDM03] who have developed an ontology for a different domain but for a similar purpose and found that a smaller ontology enhances usability.

3. **Ontology focus** Following the results of [PB88], that IBIS should be enhanced with domain specific knowledge, the developed ontology should be particularly well suited for ontology design.

4. **Adaptivity** The *Argumentation Ontology* should allow for capturing the structure of argumentation. Hence, the design must take into account that e.g. humans discuss on a free text basis while ontology learning algorithms use formal, structured and detailed reasons for different proposals.

5. **Support entire argumentation** The *Argumentation Ontology* should support the full argumentation cycle. This includes issue raising, conflict mediation, bargaining, clarification and agreement. Participants should be aware of which issues are currently under discussion, postponed, agreed and discarded.

6. **Conceptual as well as formalization level** People might agree on the need for a certain conceptual model but not on its actual implementation. The model should support argumentation on both conceptual and formal models.

7. **Modularization** Although the ontology should support the ontology engineering process we do not aim to support every part of it. As described in [SS02] the ontology engineering process involves the definition of requirements, owners and other meta attributes like Dublin core meta data. These should not be modelled here.

8. **Formalism independence** The *Argumentation Ontology* should be independent of the formalism used to model the final ontology. Each formalism allows different sets of modelling decisions and all can be subject to discussion. However, the formal model of the finally agreed ontology should be a result of the instantiation of the *Argumentation Ontology*.

9. **Process awareness** The *Argumentation Ontology* is embedded into the DILIGENT process presented in section 4.4. Essential properties of this process are its collaborative aspects, its distributiveness and the asynchronous way participants can provide arguments.

10. **Argumentation formalization** Although we do not currently plan to provide the arguments themselves in a formal way, the *Argumentation Ontology* should allow us to do so. As our last use case has illustrated ontology engineering and ontology usage could gain from such a formalization.

4.7.4 Argumentation Ontology Description

The DILIGENT *Argumentation Ontology* is visualized in figure 4.8[14]. The main concepts in our ontology are issues, ideas and arguments, which are represented as classes. These are in line with the terminology proposed by the IBIS methodology (req. 1). Issues introduce new topics in the discussion from a conceptual point of view. They are used to discuss what should be in the conceptual model of the ontology without taking into account how these items should actually be formalized and implemented in the ontology (req. 8). Ideas refer to how these concepts should be formally represented in the ontology,

[14]The corresponding OWL ontology is available online at `http://diligentarguonto.ontoware.org/`.

for instance as a class, an instance, etc. They relate to concrete ontology change operations[15]. Ideas are related to issues in the sense that they *respond to* them. Ideas refer to how issues should actually be implemented in the ontology. Therefore, "respond to" is a relation between an Idea and Issue. In this way discussions can take place in both the conceptual level and the formalization level (req. 6). Arguments are *argument*s *on* either one particular idea or one particular issue. Typically, our domain experts will start by proposing new issues to be introduced in the ontology. Arguments will be exchanged over them. Then, they discuss how these issues should be formalized through concrete ideas. Domain experts can also provide elaborations. These are issues that refine an issue under discussion, *elaborates on*.

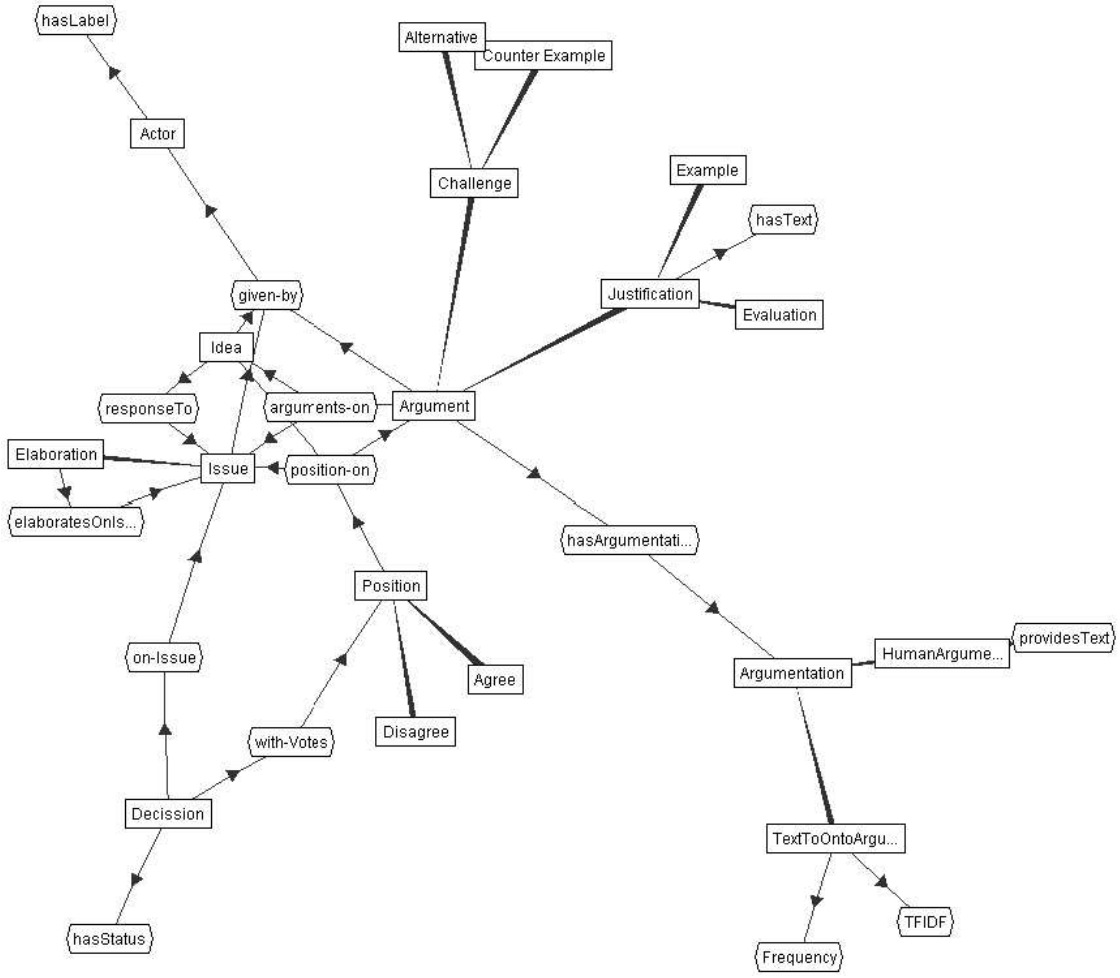

Figure 4.8: The major concepts of the argumentation ontology and their relations

Since concepts to be represented in an ontology should be consensual, this requires some consensus building discussions. In DILIGENT processes, concepts are only added to the ontology if they can be agreed upon, that is after some arguments have been ex-

[15]For example [SMMS02] presents a formal model for ontology change operations.

changed, positions by different actors have been issued on them and some decisions have been made. Arguments for (pro) an idea or issue are called justifications. Arguments against (con) an idea or issue are called challenges. In what regards arguments in favor, particularly useful OE processes, we identified examples and evaluation&justification. Two classes in challenges are also particularly used in OE discussions: counter examples and alternative&contrast. These arguments focus the IBIS argumentation methodology for Ontology Engineering (req 3).

Those involved in discussions can state positions. They clarify the *position on* one issue, one idea, or an argument under discussion. Either one agrees or disagrees. Once enough arguments have been provided and positions have been stated on them decisions can be made. In general, positions lead to decisions. Decisions are taken on issues. A decision has a status that can vary from under-discussion, postponed, discarded and agreed (req 5). A decision records not only the *issue on* which it was taken, but also both the positions issued when final *with-votes* (several positions) were cast and the *line of reasoning* (a sequence of arguments) underlying the decision on that issue. A decision can also state the idea *on-idea* underlying its issue. This allows one to focus on the relevant arguments (req 2).

Arguments are *given by* actors (req 9). We can have different kinds of Actors: either Humans or Machines. Different kinds of actors provide different argumentations (req. 4). In what regards argumentation humans (HumanArgumentation) tend to argue by providing strings of text stating (provides text) their reasons while machines tend to use other kinds of argumentation measures like Frequency and TFIDF [MS01]. For each algorithm used, new subclasses of argumentation need to be introduced to model the different kinds of measures.

4.7.4.1 Example: An Argumentation Ontology for DILIGENT Processes

The following discussion transcript was a part of an experiment performed at our institute (*cf.* [PSTS04], section 4.6.2). The participants were asked to build an ontology for modelling the research interests of our group. The experiment lasted for 90 min. and involved eleven actors. The participants provided their arguments in free text without formal restrictions. Hence, in the following example we model the discussion ex post. Moreover, we do not aim to model the entire discussion, but pick out an excerpt to exemplify our model.

> ...
> **cs:** We have done quite a bit of research in distributed knowledge management (DKM) lately. So I suggest DKM as a topic plus a subtopic "peer to peer" (P2P)

The actor suggests on the one hand to introduce "DKM" and "P2P" in the ontology (Issues), and proposes on the other hand to model them as "topics" (Ideas).

Formalization
Individual(issue1 type*(*Issue*)* value*(*states *"I suggest DKM"))*
Individual(issue1 type*(*Issue*)* value*(*given-by *actorCS))*
Individual(justi1 type*(*Justification*)* value*(*hasArgumentation *argumentation1))*
Individual(justi1 type*(*Justification*)* value*(*arguments-on *issue1))*
Individual(argumentation1 type*(*HumanArgumentation*)* value*(*providesText *"We have ... lately"))*
Individual(idea1 type*(*Idea*)* value*(*respondsTo *issue1))*
Individual(idea1 type*(*Idea*)* value*(*ontoChange *add(DKM:Topic)))*
Individual(elaboration2 type*(*Elaboration*)* value*(*states *"P2P subtopic DKM"))*
Individual(idea2 type*(*Idea*)* value*(*respondsTo *elaboration2))*
Individual(idea2 type*(*Idea*)* value*(*ontoChange *add(DKM supertopic P2P)))*

> **ah:** I suggest knowledge management (KM) as super concept of DKM because every DKM is a kind of KM

The second actor agrees implicitly with the suggestion to introduce "DKM" in the ontology. In contrast to the first one he proposes to model it as a "concept".
Formalization
...
Individual(idea3 type*(*Idea*)* value*(*ontoChange *add(KM:Concept)))*
...

> **jt:** Well I am now wondering whether P2P is DKM, because File exchange is not always KM is it?

A third actor agrees also implicitly, that "P2P" and "DKM" are important for the domain, but challenges that they should be modelled in the proposed way.
Formalization
Individual(counter1 type*(*CounterExample*)* value*(*hasArgumentation *argumentation3))*
Individual(counter1 type*(*CounterExample*)* value*(*arguments-on *elaboration2))*
Individual(argumentation2 type*(*HumanArgumentation*)* value*(*providesText *"File exchange ... KM"))*

> **ph:** I suggest Distributed Comp. (DC) with P2P and Grid as subtopics; DKM as subtopic of DC and KM

The fourth actor presents a new issues which could resolve the conflict.
Formalization
...
Individual(issue2 type*(*Issue*)* value*(*states *"I suggest DC"))*

Individual(elaboration3 type*(*Elaboration*)* value*(. . . . .))*
. . .

do: PRO **ph** : because his approach separates KM and distributiveness

The actor "do" agrees with the suggestion and provides additional reasons for the design. Implicitly he also agrees that "KM" should be part of the ontology.
Formalization
Individual(position1 type*(*Agree*)* value*(*position-on *elaboration3))*
. . .

cs: I'd like to agree to **ph** and **do** suggestion.
. . .

The first actor agrees with the new solution to introduce DKM as subclass of KM and DC and discards his original proposal to introduce P2P as subtopic of DKM.

This example demonstrates that OE discussion can be modelled with the DILIGENT *Argumentation Ontology*. The applicability of the ontology will depend on the available tool support. We do not intent to automatically annotate a free discussion. We rather envision a template based approach to provide arguments. Currently we use a WIKI to support the argumentation process. However, integration with reasoners and inclusion into existing OE environments is desirable, but remains to be done.

Chapter 5

ONTOCOM cost estimation model

5.1 Introduction

Ontology Engineering is currently advancing from a pure research topic to real applications. This state of the art is emphasized by the wide range of European projects with major industry involvement and, in the same time, by the ever-growing interest of small and medium size enterprizes asking for consultancy in this domain. A core requirement in all of these efforts is, however, the availability of proved and tested methods which allow an *efficient* engineering of high-quality ontologies, be that by reuse, new building or automatic extraction methods. Several elaborated methodologies, which aid the development of ontologies for particular application requirements, emerged in the last decades. Nevertheless, in order for ontologies to be built and deployed at a large scale, beyond the boundaries of the academic community, one needs not only technologies and tools to assist the engineering process, but also means to *estimate and control its overall costs*. These issues are addressed only marginally by current ontology engineering approaches though their importance is well recognized in the community.

A first attempt to bridge this gap has been made with the ONTOCOM approach to estimate costs[PBM05b, PBM05a], which is intended to be used as a method to estimate the efforts involved in building, reusing and maintaining ontologies. Likewise in the adjacent field of Software Engineering, a discipline in which cost prediction models belong to standard development environments, ONTOCOM proposes a top-down, parametric cost estimation method on the basis of pre-defined process stages and cost drivers, whose impact has been derived in advance by taking into account research and historical data from previous projects.

ONTOCOM distinguishes among three top-level phases of an ontology engineering process: 1). the development of a new ontology from scratch in conjunction with 2). reusing existing ontological sources and 3). the maintenance of ontologies in form of insertions, deletions and modifications of the initial content. For these categories cost drivers influencing the required development effort (in terms of person months) have been

identified on the basis of a comprehensive analysis of current engineering methodologies and related case studies. Every cost driver is associated with effort multipliers (from very high to very low), depending on the individual characteristics of the application. A first estimation of the importance factors was performed based on ex post analysis of different ontology engineering efforts and preliminary expert validations with very promising results. Nevertheless more empirical data as well as an in-depth model validation and refinement are essential requirements for the elaboration of a reliable cost estimation method. The validation of the model [PBM05a] is based on the quality framework for cost prediction by Boehm [Boe81], which is a list of required and desirable features for these category of models, showing many similarities with established frameworks for evaluating the quality of general purpose information or data models [Epp01].

For the DILIGENT methodology we have decided to incorporate the ONTOCOM cost estimation model into the methodology. In the following we describe our experiences in the application of ONTOCOM to the DILIGENT methodology. This attempt was motivated by the results of the above mentioned preliminary validation of the cost model, which made clear that the usability of the model would be significantly improved if we directly associate the cost drivers to more specific sub-tasks of the ontology engineering process, because this alignment would enable the definition of a more precise and in the same time more reliable data extraction procedure. DILIGENT offers a fine-grained description of steps and activities required to collaboratively develop ontologies in application scenarios which have to cope with both distributed ontology usage and evolution, and volatile requirements arising during its life cycle. For this purpose it covers the most important process stages and activities which are well-recognized to be part of every typical ontology engineering scenario, thus being an excellent candidate for the refinement of the (yet very high-level) cost estimation methodology. On the other hand, DILIGENT currently lacks any support to estimate the effort required by the corresponding engineering process. The decision to exit a certain stage and enter a new cycle of the process model is not supported adequately. For this reason, a second goal of aligning ONTOCOM to the engineering methodology was to extend the latter with a cost estimation dimension in order to examine the ways costs could be involved as a decision support criteria to justify transitions between various process stages.

The remaining of this chapter is organized as follows: after introducing the ONTOCOM cost model in Sections 5.2, we describe the refinement of the cost model towards applying it to the DILIGENT process model and the changes triggered by this mapping (Section 5.3). The results of this mapping are used to specify the cost prediction procedure for concrete DILIGENT case studies in Sections 5.4 and 5.5. An overview of related and future work are given in Sections 2.5.3 and 5.6, respectively.

5.2 The ONTOCOM Cost Model

ONTOCOM is a parametric cost estimation model for ontologies which aims at predicting the effort invested in building, maintaining and reusing ontologies on the basis of pre-defined cost drivers. For the definition of the relevant factors, we adopted a combination of three general-purpose cost estimation methodologies [Boe81], which are in our opinion applicable to Ontology Engineering according to the current state of the art in the field (see [PBM05a] for an overview of the examined methods). We started with a top-down approach to identify the cost drivers, by identifying upper-level sub-tasks of the ontology engineering process and defining the associated costs using a parametric method in correlation with a human-driven first validation. Expert judgement was used to evaluate the set of cost drivers associated to each process stage and to specify their start values in the a-priori model. Initially, we distinguished among three areas, whose costs were to be defined separately:

Ontology Building includes the typical stages of an ontology engineering process[FL99]: domain analysis (result: requirements specification), the conceptualization (result: conceptual model), the implementation (result: specification of the conceptual model in the selected representation language) and the ontology population i.e. the generation of instances and their alignment to the model (result: instantiated ontology).[1]

Ontology Maintenance includes getting familiar with and modifying the ontology (insert or delete new ontological primitives, re-model parts of the ontology etc.)

Ontology Reuse involves costs for the discovery and re-usage of existing (source) ontologies in order to generate a new (target) ontology. The latter includes understanding, evaluating and adapting the former ones to the requirements of the latter.

This upper-level distribution is of course subject of future refinements in order to increase the usability of the estimation method in real-world engineering projects. In particular, the ontology building area should be elaborated in the same top-down manner in order to partition this tedious and complex process down to a level in which the associated efforts can be reliably predicted. In this case, the cost drivers relevant the overall ontology building process (see below) are to be aligned (or even re-defined) to the corresponding sub-phases and activities. In particular, this issue will be further discussed in Section 5.3, where we describe the revision of the cost drivers related to building and reusing ontologies in conjunction with the DILIGENT framework.

Starting from a typical ontology engineering scenario, in which an ontology is developed—from scratch, by adapting existing knowledge sources, or both—and de-

[1] At this point we restrict to manual ontology building activities. Automatic ontology generation methods as those proposed in the area of ontology learning are not considered in this work yet.

ployed/maintained by its users, ONTOCOM calculates the necessary person-months effort using the following equation:

$$PM = PM_B + PM_M + PM_R \tag{5.1}$$

PM_B , PM_M and PM_R represent the person months associated to building, maintaining and reusing ontologies, respectively and are calculated as:

$$PM_X = Size_X * \prod CD_{Xi} \tag{5.2}$$

Each of the three development phases is associated with *specific* cost factors. Experiences in related engineering areas [Kem87, Boe81] let us assume that the most significant factor is the *size of the ontology* involved in the corresponding process or process stage. In the formula above the size parameter $Size_X$ is expressed in thousands of ontological primitives – concept, relations, axioms and instances.[2] $Size_B$ corresponds to the size of the newly built ontology i.e. the number of primitives which are expected to result from the conceptualization phase. In case of ontology maintenance the size of the ontology ($Size_M$) depends on the expected number of modified items. For reuse purposes the relevant factor $Size_R$ is the (total) size of the original source(s) after being tailored to the present application setting. In particular this involves the parts of the source ontologies which have to be translated to the final representation language, the ones whose content has to be adapted to the target scope and the fragments directly integrated. The *cost drivers* CD_{Xi}—where X stays for B, R and M, respectively—have a rating level (from very low to very high) that expresses their impact on the development effort. For the purpose of a quantitative analysis, each rating level of each cost driver is associated to a weight (effort multiplier - EM). In the a-priori cost model a team of 3 ontology engineering experts assigned start values between 0.7 and 1.9 to the effort multipliers, depending on the perceived contribution of the corresponding cost driver to the overall development costs.[3] These values are subject of further calibration on the basis of the statistical analysis of real-world project data. Additionally the values of the a-priori model (i.e. containing non-calibrated values) will be included in the expert validation process, which will is currently being performed as part of the mapping between ONTOCOM and the engineering methodology DILIGENT.

In the following we turn to a brief description of the cost drivers in ONTOCOM.[4] These parameters were derived after surveying recent literature and from empirical findings of various case studies in the ontology engineering field. For each cost driver we specified in detail the decision criteria which are relevant for the model user in order for him to determine the concrete value of the driver in a particular situation. For example for the cost driver LEXP—accounting for costs produced by the level of experience of the engineering team w.r.t. ontology representation languages—we pre-defined the meaning of

[2] For example for an ontology with 1000 concepts and 100 relations $Size$ will have the value 1.1.

[3] A list of the values is available in [PBM05a].

[4] See [PBM05a] for a detailed explanation of the approach to estimate costs.

the effort multipliers as depicted in Table 5.1. The values of the corresponding effort multipliers, which have been specified by human experts, are as follows: 1.60 (Very Low), 1.30 (Low), 1 (Nominal), 0.90 (High) and 0.80 (Very High) [PBM05a]. The suitable value should be selected during the cost estimation procedure and used as a multiplier in equation 5.2.

	Very Low	**Low**	**Nominal**	**High**	**Very High**
LEXP	2 months	6 months	1 year	3 years	6 years
Effort Multipliers	1.60	1.30	1.0	0.90	0.80

Table 5.1: Cost Driver LEXP (Language Experience)

In several cases the decision criteria associated with a cost driver are more complex than in the previous example and might be sub-divided into further sub-categories, whose impact is aggregated to the final value of the corresponding cost driver by means of weights.

5.2.1 Cost Drivers for Ontology Building

For the ontology building area we defined a list of cost drivers, which are, similar to [B. 97], divided into three groups:

- **Product-related** cost drivers account for the influence of ontology characteristics on the overall costs: i) Instance (DATA) to capture the effects that the instance data requirements have on the overall process, ii) Ontology Complexity (OCPLX) to express those ontology features which increase the complexity of the engineering outcomes, iii) Required Reusability (REUSE) to capture the additional effort associated with the development of a reusable ontology, and iv) Documentation match to life-cycle needs (DOCU) to state for the additional costs caused by very detailed documentation requirements.

- **Project-related** cost drivers relate the dimensions of the engineering process and its characteristics to the overall costs: i) Support tools for Ontology Engineering (TOOL) to measure the effects of using ontology management tools in the engineering process, and ii) Multisite Development (SITE) to mirror the usage of the communication support tools in a location-distributed team.

- **Personnel-related** cost drivers emphasize the role of team experience, ability and continuity w.r.t. the effort invested in the process: i) Ontologist/Domain Expert Capability (OCAP/DECAP) to account the perceived ability and efficiency of the single actors involved in the process (ontologist and domain expert) as well as their teamwork capabilities, ii) Ontologist/Domain Expert Experience (OEXP/DEEXP)

to measure the level of experience of the engineering team w.r.t. performing ontology engineering activities, iii) Language/Tool Experience (LEXP/TEXP) to measure the level experience of the project team w.r.t. the representation language and the ontology management tools, and iv) Personnel Continuity (PCON) to mirror the frequency of the changes in the project team.

5.2.2 Cost Drivers for Ontology Reuse and Maintenance

Additionally to project and personnel cost drivers (as described in Section 5.2.1) we defined a set of further 4 cost drivers to deal with the characteristics of ontology reuse and maintenance, as reported by recent case studies in these areas [PBMT05, UHW+98, RVMS99, UCH+98]:

- **Ontology Understanding(OU)** accounts for the efforts required to get familiar with the ontologies to be used, a task which is a pre-condition to ontology evaluation and maintenance. It depends on ontology properties such as representation language or size and on the level of experience of the ontology engineer w.r.t. this ontology[PBM05a].
- **Ontology Evaluation(OE)** accounts for the additional efforts related to the evaluation phase given a satisfactory ontology understanding level (e.g. for testing the source ontologies against a specific set of requirements).
- **Ontology Modification/Translation(OM/OT)** are factors reflecting the costs involved by adapting the source ontologies to the new setting (e.g. inserting, deleting ontology concepts) and by translating to a target representation language, respectively.

5.2.3 Evaluation of ONTOCOM

The parametric approach to estimate costs for OE processes described in this report is currently being validated towards a reliable method for estimating the costs of ontology engineering. The most important evaluation criterium is of course the reliability of its predictions, which however depends on the amount of accurate project data used to calibrate the model (i.e. adjust the values of the modifiers and identify eventual correlated cost drivers). On the other hand, a comprehensive evaluation of the model should go beyond the evaluation of its functionality (i.e. the accuracy of its estimations) and also address issues related to its usability in typical ontology engineering scenarios, as suggested in common quality frameworks for information systems (such as [PS04, MSBS03, HLW99, KLS95]; see [Epp01] for a comprehensive survey on this topic).

For a comprehensive evaluation of the model we rely on the quality framework for cost models by Boehm, which was adapted to the particularities of ONTOCOM and Ontology

Engineering. Parts of this framework are used in to assess the quality of the a-priori and the a-posteriori cost models, respectively (see below). According to this differentiation, the evaluation of the cost model is performed in two steps: in the first one we evaluate the relevance of the mentioned cost drivers for the purpose of predicting costs arisen in ontology engineering projects; the remaining aspects of the framework relate to its capability of reliably fulfilling its goal (i.e. that of estimating engineering efforts) and will be applied in a second step on the a-posteriori model resulting from the calibration of the preliminary one.

The original quality framework by Boehm [Boe81] consisted of the 10 features, which we divided into two categories, depending on their relevance to the a-priori and the a-posteriori model, respectively. The meaning of the quality criteria has been adapted to the scope of ONTOCOM.

A-priori evaluation :

definition : has the model clearly defined the costs it is estimating and the costs it is excluding? Does the estimate include the cost of management, training, domain analysis, conceptualization, implementation, testing, maintenance? Does the model clearly define the decision criteria used to specify the ratings of the cost drivers? Does the model use intuitive and non-ambiguous terms to denominate the cost drivers it involves?

objectivity : does the model avoid allocating most of the cost variance to poorly calibrated subjective factors? Are the cost drivers defined using objective decision criteria, which allow an accurate assignment of the corresponding cost driver ratings?

constructiveness can a user tell why the model gives the estimates it does?

detail : does the model easily accommodate the estimation of new process models or is it conceived for a particular ontology engineering process? Does it give accurate phase and activity breakdowns? Does the model take into consideration factors related to the main tasks of the engineering process? Do these sub-tasks correspond to the process model applied in your engineering process? Which phases should be further covered by the model in order to increase its usability?

stability : do small differences in inputs produce small differences in output cost estimates?

scope : does the model cover the class of projects whose costs you want to estimate? Is it representative for a wide class of ontology engineering projects?

ease of use : are the model inputs and options easy to understand and specify? Is it easy for you to assess a rating to a cost driver based on the associated decision criteria?

prospectiveness : does the model avoid the user of information which will not be well known until the project is complete? Can the model be applied in early phases of the project as well?

parsimony : does the model avoid the use of highly redundant factors or factors which make no appreciable contribution to the results?

A-posteriori evaluation :

all items of the former category, plus

fidelity , since this requirement will definitely not be fulfilled after collecting reliable data from previous projects used to refine the values of the cost drivers and to discover eventual correlations between them.

The features assigned to the first category will be used as criteria for the refinements resulting from the application of the cost model to DILIGENT, as described in Section 5.3.

5.2.4 Current Limitations

As mentioned above the approach to estimate costs for OE processes we described in this section is intended as a first draft towards a reliable cost estimation method for ontology engineering projects. The relevance of the cost drivers and their quality in terms of scope and granularity will be assessed during the first step of the model evaluation procedure. In particular the factors describing the development of new ontologies should be further refined in order to realistically reflect the efforts invested in ontology building, which is well-recognized as a tedious and challenging process. Mapping these cost drivers to a fine-grained process model such as that proposed by the DILIGENT methodology is an excellent opportunity to prove their expedience in the described context. Finally the usability of the model is directly related to the collection of real-world project data, which are of course indispensable for the calibration of the values required for the parametric prediction equation. The case studies currently being carried out according to the DILIGENT methodology will provide valuable information w.r.t. this second issue.

5.3 Bridging ONTOCOM and DILIGENT

The main objective of this section is to integrate the mentioned cost model ONTOCOM to ontology engineering processes performed according to the DILIGENT engineering methodologies. Mapping the two models is mainly aimed at validating and refining the existing cost estimation strategy on the basis of real-world ontology development case studies. In order to increase the usability of the cost model and to collect high quality data necessary for its calibration we examined the relevance of the pre-defined cost

drivers w.r.t. the process sub-tasks and actions and precisely define the ways the data is to be collected. This task resulted in an adaptation of the estimation model and the process model since missing parts could be identified. Further on we adapted the high-level parametric effort estimation equation from ONTOCOM to the DILIGENT scenario in order to be able to apply the model to the case studies currently being performed in the basis of this engineering methodology. The necessary data is to be collected from the case study partners in the near future and will provide first estimations of the envisioned total development costs and revise the cost model. The calibrated cost estimation model will allow its users to estimate in each process stage the future effort necessary to build and maintain the ontology. Cost-related information might be a significant decision factor for obtaining a feasible trade-off between start-up/maintenance efforts and ontology utility. The last part of this section reports on our experiences in trying to determine optimal relation between the initial effort, the utility and the maintenance effort implied by a concrete DILIGENT-based ontology development project.

To summarize, bridging the two models was performed according to the following steps, which will be further elaborated in the remaining of this section:

1. Alignment of the cost drivers with the DILIGENT process stages
2. Definition of the cost function and of data collection procedure in terms of the process model
3. Examination of the potential process decision support in terms of costs

The data collection and the subsequent calibration of the cost model according to the outcomes of this task will be performed in the near future in terms of the procedure described below.

5.3.1 Mapping the Activities in DILIGENT to the Cost Drivers in ONTOCOM

In order to realistically estimate the costs induced by particular DILIGENT processes the general cost model introduced in 5.2 has to be customized to the activities and the stages of the engineering methodology. As introduced in section 4.4 the DILIGENT process is divided into five main stages, where each stage contains a number of sequential or parallel activities. The ONTOCOM to DILIGENT mapping assigns product, personnel and project cost drivers to the corresponding engineering activities. As mentioned earlier we use the general description of DILIGENT for this mapping. Tables 5.2 to 5.5 summarize the results of this mapping (for simplification purposes the costs associated to the analysis and revision phases were joined to a single overview table). Note that the changes occurred at the cost model level as a result of its alignment to the engineering methodology have been printed in the tables below in italics.

5.3.1.1 Cost Drivers Relevant for the Centralized Building Stage

DILIGENT process		**Cost factor**		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Centralized Building	Domain analysis	Complexity of the domain analysis	OCAP, DECAP OEXP, DEEXP TEXP, PCON	TOOL
	Conceptualization and implementation of shared ontology	*Complexity of the conceptualization (DOCPLX)*, REUSE, OU, OE, OT, OM, *Ontology Integration (OI)*	OCAP, DECAP OEXP, DEEXP, LEXP, TEXP PCON	TOOL
	Evaluation of shared ontology	*Ontology evaluation (OE)*	OCAP, DECAP OEXP, DEEXP, LEXP, TEXP, PCON	TOOL
	Documentation	DOCU	OCAP, DECAP OEXP, DEEXP, TEXP, PCON	TOOL

Table 5.2: Mapping DILIGENT to ONTOCOM: Centralized Building

The calculation of the expected costs for the centralized building stage (*cf.* table 5.2) depends on all cost drivers except the ones for Ontology Maintenance and Required Development Schedule (SCED). We can leave out the cost driver for maintenance as it is part of subsequent stages to keep the ontology updated. The cost driver SCED is left out since we are currently only interested in the estimation of person month rather than the complete project duration. This applies also for the subsequent process stages.

The cost drivers for ontology reuse should be considered only if the board decides to utilize existing ontologies to build the initial shared ontology.

We found out that the initial proposal in the DILIGENT process model to reduce the centralized building to only one activity was too coarse. Hence, we introduced four activities to capture this phase more accurately. Splitting up the building process into those activities led us to the conclusion that the cost driver for ontology complexity was too broadly defined. As a consequence, this cost driver was divided into three separate new ones, namely one for the complexity of the domain analysis, one for the complexity of the conceptualization and finally one for the implementation complexity.

Similarly, the cost model so far did not define any cost driver for the integration of different ontologies. From a process point of view integrating ontologies with each other

is a rather time consuming task. Furthermore the cost driver for tool support covered only the tools available to formalize the ontology. Experience suggests, that for each activity different supporting tools are available. The cost driver for tool support should therefore be interpreted as an average of the quality of the available tools for each activity. Finally, the evaluation process for the resulting ontology was not covered by any cost driver.

Even though the process defines a separate activity for the provision of arguments, we concluded that no extra costs driver is needed to calculated the respective effort, because it is already covered by the documentation cost driver.

5.3.1.2 Cost Drivers Relevant for the Local Adaptation Stage

The cost drivers relevant to estimate the costs of the local adaptation stage (*cf.* table 5.3) are to a large extent the same as those associated to the previous stage. While the personnel factors relate to the capabilities of the domain experts, the main difference of this stages compared to the Centralized Building one is the application of the cost drivers to a multitude of sites and ontologies, while different users belonging to different sites are involved in different activities performed on the shared, the locally adapted and the externally adapted ontologies respectively. The sizes of these ontologies are put into relation with different groups of cost drivers when considering the calculation of the resulting costs. All users follow the process defined in the Local Adaptation stage, therefore each of them incurs some effort to understand, adapt and use the ontology. As the usage and customization of the shared ontology is theoretically inconceivable without it being previously understood by its users, one major cost category of the Local Adaptation phase is of course related to the Ontology Understanding effort multiplier in conjunction with the size of the currently shared ontology. Further on the modification of the shared ontology depends on the number of changes performed together with the corresponding effort multipliers for ontology maintenance in ONTOCOM. Additionally to the original cost model DILIGENT foresees an activity in which new requirements are specified, which is mapped to a cost driver in the ontology building category. If the users decide to adopt externally developed ontologies to conceptualize their local changes (i.e. instead of newly implementing them in their local ontology) the corresponding activities imply additional costs, as described in the ontology reuse category in ONTOCOM. Note that there are no costs for the translation of the ontologies to be reused since it is assumed that all ontologies in the application scenario are formalized in the same representation language. When calculating the cost estimation for the adaptation phase we recommend to use average values for the aforementioned effort multipliers instead of indicating separate values for each site.

5.3.1.3 Cost Drivers Relevant for the Centralized Analysis and Revision Stage

From an cost estimation point of view the centralized analysis and revision stage (*cf.* table 5.4) resembles the building stage, while the former involves both the currently shared

and the locally used ontologies. Each of these ontologies is associated with different categories of cost drivers. As the original shared ontology is already known to the board, there are no additional costs to understand it. The board examines in turn the changes submitted by the users. The number of SITEs, as well as the average number of changes

DILIGENT process		**Cost factor**		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Local adaptation	Local analysis of shared ontology	OU, OE	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL
	Specification of new requirements	*Complexity of the domain analysis (DOCPLX)*	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL
	Ontology utilization		DECAP, DEEXP, LEXP, TEXP	TOOL
	Ontology instantiation	DATA	OEXP, DEEXP, OCAP, DECAP, TEXP, LEXP, PCON	TOOL
	Local analysis of additional (local) ontologies	OU, OE, number of sites	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Customization of relevant local ontologies	OM, number of sites	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Integration of reused local ontologies to the shared ontology	*Ontology integration (OI)*	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Modification of shared ontology Argument provision	OM	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Evaluation of new local ontology	*Ontology evaluation (OE)*	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Documentation	DOCU	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL

Table 5.3: Mapping DILIGENT to ONTOCOM: Local Adaptation

introduced by the users will thus influence the costs. In the analysis and revision stage all requirements derive from user requests, thus we do not incur costs for domain analysis. The local ontologies from the users are available and no search effort is needed to obtain them. The costs for ontology modification depend on the actual changes introduced by the board.

W.r.t. the personnel cost drivers we need to distinguish between the average experience of the users and the experience of the board, while the latter is rated higher than the former when it comes to ontology engineering.

5.3.1.4 Cost Drivers Relevant for the Local Update Stage

For the last stage in the DILIGENT process, the Local Update (*cf.* table 5.5) we need less cost drivers than for the previous stages. The relevant sizes are the number of changes introduced by the board and the number of sites. As in previous stages the personnel and management cost drivers remain unchanged. We have identified the three product cost drivers Ontology Understanding, Ontology Evaluation and Ontology integration as relevant for this stage.

5.3.2 Changes in the Cost Model

Applying the pre-defined ONTOCOM cost drivers to the DILIGENT engineering process model revealed several shortcomings of the former w.r.t. two dimensions: the insufficient coverage of some of the available cost drivers and the lack of support for tasks such as ontology merging and integration. Additionally to the adjustment of the model according to these findings, the expert values used by the a-priori cost estimation formula were adapted on the basis of the expertise provided by the DILIGENT engineering team.

We now turn to a detailed description of the model refinements arisen during the integration of the two models.

5.3.2.1 The Cost Driver OCPLX (Ontology Complexity)

The mapping between the DILIGENT process stages and the available product-based cost drivers made clear that the original cost driver coping with the complexity of the product to be developed (i.e. the ontology) was not covering all aspects of the ontology building process to a satisfactory extent. As already assumed by the authors, from a product perspective, distinguishing between three high-level phases (building, reuse and maintenance) has proved to be insufficient for the needs of real-world settings. Though the OCLPX cost driver, which was initially intended to cover the costs arisen from this activity, already mentioned some of the most important factors in this field, the DILIGENT ontology engineers evaluated its impact in the current form as too limited compared to

the impact of other, less relevant and less complex cost drivers such as REUSE or DOCU (see [PBM05a] for a detailed description of the cost drivers). Consequently we divided the original OCPLX cost driver into three new parameters partially covering the overall complexity of the target ontology. The division in three complexity areas was performed

DILIGENT process		**Cost factor**		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Centralized Analysis and Revision	Information collection from users			SITE
	Analysis of obtained information	OU, OE, number of sites	OEXP, OCAP, LEXP, TEXP, PCON	TOOL
	Control of previously shared ontology		OEXP, OCAP, TEXP, PCON	TOOL
	Specification of new requirements		OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	Customization of relevant local ontologies	OM, number of sites	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Integration of reused local ontologies to the shared ontology	*Ontology integration (OI)*	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Modification of shared ontology	, OM, REUSE	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	Argument provision			
	Argumentation aggregation			
	Evaluation of new shared ontology	*Ontology evaluation (OE)*	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	Documentation	DOCU	OCAP, OEXP, TEXP, PCON	TOOL
	Distribution of new shared ontology			SITE

Table 5.4: Mapping DILIGENT to ONTOCOM: Centralized Analysis and Revision

at the process level. This design decision is justified through the assumption that the complexity of the final ontology is implicitly related the complexity of the underlying building process, in particular the phases domain analysis, conceptualization and implementation, given a certain competence level of the personnel and sufficient project experience. The decision criteria for assigning a specific rating level to the new cost drivers were mainly derived from the ones originally proposed for the OCLPX driver. Additionally new criteria related to the availability of useful knowledge sources during the domain analysis phase and to the ontology implementation were introduced (see below). We now turn to a description of the new complexity parameters:

- Domain complexity (DCLPX)
- Conceptual complexity (CCPLX)
- Implementation complexity (ICPLX)

The domain complexity driver states for the efforts additionally arisen in the engineering project by the particularities of the ontology domain and its analysis during ontology building. The decision which concepts will be included and in which form they will be represented in an ontology depends not only on the intrinsic domain to be modeled (e.g., tourism), but rather on the application domain. The latter also involves the technical setting and the characteristics of the application in which the ontology is designed to be integrated to. As a third decision field we introduced the sources which could be eventually used as additional domain descriptions and thus as an aid for the domain analysis and the subsequent conceptualization. The global value for the DCLPX driver is a weighted sum of the aforementioned areas, which are depicted in Table 5.6.

In order to realistically classify the complexity of the domain analysis phase in terms of the pre-defined ratings we identified characteristics of the three areas which usually

DILIGENT process		**Cost factor**		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Local update	Control of new shared ontology		DECAP, DEEXP, TEXP	TOOL
	Local analysis of changes in the new shared ontology	OU, OE	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL
	Integration of new and old version	*Ontology integration (OI)*	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL

Table 5.5: Mapping DILIGENT to ONTOCOM: Local Update

Rating Scale	DOMAIN
Very Low	narrow scope, common-sense knowledge, low connectivity
Low	narrow to moderate scope, common-sense or expert knowledge, low connectivity
Nominal	moderate to wide scope, common-sense or expert knowledge, moderate connectivity
High	moderate to wide scope, common-sense or expert knowledge, high connectivity
Very High	wide scope, expert knowledge, high connectivity
	REQUIREMENTS
Very Low	few, simple req.
Low	small number of non-conflicting req.
Nominal	moderate number of req., with few conflicts, few usability req.
High	high number of usability req., few conflicting req.
Very High	very high number of req. with a high conflicting degree, high number of usability req.
	INFORMATION SOURCES
Very Low	high number of sources in various forms
Low	competency questions and text documents available
Nominal	some text documents available
High	some unstructured information sources available
Very High	none

Table 5.6: The Domain Complexity Cost Driver DCLPX

influence this measure. For the domain category, we considered the scope (narrow, moderate, wide), the commonality of the knowledge (be that common-sense knowledge or expert knowledge) and the connectivity of the domain. The latter is expressed in the number of interdependencies between domain concepts with ranges again among three levels (low, moderate and high), while the scope is a feature which is related to the generality, but also to the perceived amount of knowledge comprised per default in a certain domain. For example a domain such as some department of an organization is considered narrower than a domain describing a university, while the scope of the economics domain is of course classified as wide. The three criteria are prioritized according to common practices in the ontology engineering area, so that the connectivity of the domain is considered decisive for establishing the rating of this cost factor.

The complexity of the requirements which are to be taken into consideration when building an ontology is characterized here by the total number of requirements available in conjunction with the rate of conflicting ones and the rate of usability requirements, since the latter are seen as a fundamental source of complexity for the building process.[5]

Finally the availability of information sources guiding the engineering team during the building process or offering valuable insights in the domain to be modeled can be

[5]Usability requirements express the constraints imposed by a particular characteristics of the ontology user community w.r.t. its content or content representation.

Rating Scale	Conceptualization
Very Low	concept list
Low	taxonomy, high number of patterns, no constraints
Nominal	properties, general pattern available, some constraints
High	axioms, few modeling pattern, considerable number of constraints
Very High	instances, no patterns, considerable number of constraints

Table 5.7: The Conceptualization Complexity Cost Driver CCPLX

Rating Scale	Implementation
Low	The semantics of the conceptualization compatible to the one of the impl. lang.
Nominal	Minor differences between the two
High	Major differences between the two

Table 5.8: The implementation complexity cost driver ICPLX

a major success factor in ontology engineering. When deciding upon the impact of the information sources on the effort required to perform the domain analysis activity we suggest considering the number, the type and the form of the sources.

The conceptualization complexity accounts for the impact of the structure of the conceptual ontology (taxonomy, conceptual graph etc.) and of help techniques such as modeling patterns on the overall engineering costs. On the other side, the existence of certain naming and modeling constraints might cause cost increases (see Table 5.7).

As mentioned in [PBM05a] one of the basic assumptions in ONTOCOM is that the most significant factor for estimating the costs of ontology engineering projects is the size of the conceptual model, while the implementation issue is regarded to be a matter of tools, since a manual encoding of a conceptualization in a particular formal representation language is not common practice. However the original ONTOCOM model did not pay any attention to the semantical differences between the conceptual and the implementation level, differences which might appear in situations in which the usage of a specific representation language is mandatory. In this case the implementation of the ontology requires a non-trivial mapping between the knowledge level of the conceptualization and the paradigms beyond the used representation language. The costs arisen during this mapping are stated in the driver ICPX (implementation complexity), whose ratings are illustrated in Table 5.8. For simplification reasons we restricted the range of the ratings to 3 (from low to high).

To summarize the complexity of the target ontology in ONTOCOM is taken into account by means of three cost drivers, associated with the efforts arisen in the domain analysis, conceptualization and implementation phase. We analyzed features which are responsible for cost increases in these fields – independently of the size of the final on-

tology, the competence of the team involved or the setting of the current project – and aligned them to ratings from very low to very high for quantification purposes.

5.3.2.2 The Cost Driver DOCU (Documentation Needs)

The DOCU measure is intended to state the additional costs caused by detailed documentation requirements. Likewise COCOMOII we differentiate among 5 values from very low (many lifecycle needs uncovered) to very high (very excessive for lifecycle needs) as illustrated in Table 5.9. In the original cost model DOCU was defined as a building cost

	Very Low	**Low**	**Nominal**	**High**	**Very High**
DOCU	many LC needs uncovered	some LC needs uncovered	right-sized to LC needs	excessive for LC needs	very excessive for LC needs

Table 5.9: Ratings for Documentation Costs

driver. During the model evaluation during its integration to the DILIGENT framework this driver was found to be relevant to top-level phases distinguished by the model i.e. also for reuse and maintenance.

5.3.2.3 The Cost Driver OE (Ontology Evaluation)

In the current cost model ontology evaluation is only regarded as part of the reuse phase. Our mapping to the DILIGENT methodology revealed that ontology evaluation is in fact performed before reusing external ontologies, but also after building a new ontology. Hence we broadened the the scope of the evaluation factor to building and reuse while keeping most of the original meaning (Table 5.10). While in a reuse situation the effort required for the evaluation of an ontology was monitored separately as the one implied for its comprehension, in the building case the level of the cost driver is determined autonomously of other cost factors by considering the level of activity required to test a preliminary ontology against its requirements specification document and for documentation purposes.

Rating Scale	**Ontology Evaluation**
Very Low	small number of tests, easily generated and reviewed
Low	moderate number of tests
Nominal	high number of tests
High	considerable tests, easy to moderate to generate and review
Very High	extensive testing, difficult to generate and review

Table 5.10: The Ontology Evaluation Cost Driver OE

5.3.2.4 The Cost Driver OI (Ontology Integration)

The most important ONTOCOM revision arisen as a result of the mapping to DILIGENT was the definition of cost driver for ontology reuse processes, which measures the costs produced by integrating different ontologies to a common framework. The integration step is assumed to be performed on ontologies sharing the same representation language – the efforts required for this activity are covered by the OT (Ontology Translation) cost driver [PBM05a]. As criteria influencing its complexity we identified the following:

- overlapping degree among ontologies to be integrated: it is assumed that this issue is proportional to the effort required by the integration, since it is directly related to the number of mappings between ontological entities.
- type of mappings between ontological primitives: 1 to 1 mappings are more easily discovered than multiple one (1 to n or n to m)
- integration quality, in terms of precision (rate of correct mappings) and recall (rate of mappings discovered): higher quality requirements imply automatically increased efforts to perform the integration task.
- number of ontologies: it is clear that the integration effort is directly proportional to the number of sources to be integrated

According to these considerations the ratings for the OI cost drivers were defined as depicted in Table 5.11 below.

Rating Scale	Ontology Integration
Very Low	1-1 mappings, approx. 50% precision and recall required, barely overlapping, 2 ontologies
Low	1-1 mappings, approx. 60% precision and recall required, barely overlapping, 2 ontologies
Nominal	1-n mappings, approx. 70% precision and recall required, some overlapping, 2 ontologies
High	1-n mappings, approx. 80% precision and recall required, high overlapping, more than 2 ontologies
Very High	n-m mappings, approx. 95% precision and recall required, high overlapping, more than 2 ontologies

Table 5.11: The Ontology Integration Cost Driver OI

5.3.2.5 The Cost Driver TOOL (Tool Support)

In the current cost model tool support is limited to the support of the reasoning, building and maintenance activities in the process. However, product factors like domain analysis,

Rating Scale	TOOL
Very Low	High quality tool support, no manual intervention needed
Low	Few manual processing required
Nominal	Basic manual intervention needed
High	Some tool support
Very High	Minimal tool support, mostly manual processing

Table 5.12: The Tool Support Cost Driver TOOL

integration and others can and should also be supported by tools. Instead of pre-defining the types of tools which are usually involved in an engineering project independently of the process phases in which these tools come into operation, we take account of the different levels of tool support for the different phases by one general-purpose cost driver and calculate the final value as the average tool support across the entire process.

Therefore the ratings for tool support are re-defined on a more general level, as shown in Table 5.12 below.

The rating of the cost driver should be specified for each of the most prominent process phases, while the importance of the corresponding phase is expressed in terms of weights. The global TOOL value for a specific project is calculated as a normalized sum of the weighted local values. For the DILIGENT methodology one should specify the tool support level for the following sub-tasks: domain analysis, conceptualization, implementation, ontology understanding and evaluation, ontology instantiation, ontology modification, ontology translation, ontology integration and documentation.

5.4 A Cost Function for DILIGENT Processes

In Section 5.3 we described the mapping between the ONTOCOM cost model and the DILIGENT methodology, which aimed at defining the role the cost drivers listed in the former play w.r.t. the efforts invested in individual phases and activities of the latter. On the basis of this mapping and the changes triggered by this task in both models we customized the general person month equation in ONTOCOM to the particularities of DILIGENT processes. The resulting function was further simplified in order to allow the elaboration of optimization criteria in DILIGENT, which we assumed to be useful as decision support for state transitions in the incremental engineering cycle.

5.4.1 The Complete Cost Function

The general-purpose ONTOCOM equations 5.3 and 5.4

$$PM = PM_B + PM_M + PM_R \tag{5.3}$$

$$PM_x = Size_x * \prod CD_{xi} \tag{5.4}$$

which assume a linear engineering process, in which an ontology is built from scratch, by reuse or both and is further maintained by its users, were adapted to the cyclic model of DILIGENT as in equation 5.5 below:

$$PM = PM_{CB} + \sum_{i=1}^{n}(PM_{LA_i} * m_i + PM_{CAR_i} + PM_{LU_i} * m_i) * p^i, \tag{5.5}$$

where PM_{CB}, PM_{LA_i}, PM_{CAR_i} and PM_{LU_i} are the person months necessary for the initial building phase and for the local adaptations, centralized analysis and revision and local updates in cycle i, respectively. Note that i iterates over the number of cycles n and that in every cycle the number of sites participating at the process is considered through the variable m_i. Finally, we introduced the parameter p ($p > 0$) as a learning rate between consecutive cycles in the process model. Usually we can assume that $p \leq 1$, which means that the team involved in the project improves its experience level and is able to solve the same tasks more efficiently (i.e. with less costs) from one building cycle to another. However, while the positive learning rate is intended to reflect the changes occurring on the effort multiplier level between consecutive cycles, the size of the resulting ontologies (i.e. the size of the locally modified ontologies in the local adaptation phase, the size of the shared ontology obtained after a new board meeting and the size of the final locally updated one) vary from development cycle to development cycle. This observation justifies the usage of the index i in the second part of equation 5.5 for the person months variables PM_{LA_i}, PM_{CAR_i} and PM_{LU_i}, since their values are vary with the size of the ontologies involved.

We elaborated the detailed cost functions for each of the 4 enumerated process stages: centralized building, local adaptation, centralized analysis and revision and local updates.

5.4.1.1 The Costs of the Centralized Building Phase

$$\begin{aligned} PM_{CB} = {} & Size_{CBB} * \prod PROD_{CBB} * \prod PERS * TOOL + \\ & Size_{CBR} * \prod PROD_{CBR} * \prod PERS * TOOL \end{aligned} \tag{5.6}$$

The efforts required by the centralized building phase are divided into the ones invested in building a new ontology and the ones invested in reusing external ones. The product effort multipliers are as follows:

$$\prod PROD_{CBB} = DCPLX * CCPLX * ICPLX * REUSE * DOCU * OE * OI \quad (5.7)$$

$$\prod PROD_{CBR} = OU * OE * OI * OT * OM * DOCU \quad (5.8)$$

In case of the personnel factors the multiplier values is computed as in 5.9.

$$\prod PERS = OCAP * DECAP * OEXP * DEEXP * PCON * LEXP * TEXP \quad (5.9)$$

Note that the reused size $Size_{CBR}$ should be calculated as the sum over all single ontologies reused, while the reuse equation contained in 5.6 was simplified in comparison to the original one in ONTOCOM [PBM05a].

5.4.1.2 The Costs of the Local Adaptation Phase

$$\begin{aligned} PM_{LA} = {} & Size_{LAS} * \prod PROD_{LAS} * \prod PERS_{LAS} * TOOL + \\ & Size_{LAM} * \prod PROD_{LAM} * \prod PERS_{LAM} * TOOL + \\ & Size_{LAR} * \prod PROD_{LAR} * \prod PERS_{LAR} * TOOL \end{aligned} \quad (5.10)$$

The first part of the equation calculates the effort required to evaluate and use the shared ontology ($Size_{LAS}$ is the size of the shared ontology). The rest accounts for the additional efforts arisen if external ontologies (i.e. developed locally at different sites) are analyzed for being reused in the local context or if the shared ontology needs to be modified ($Size_{LAM}$ is the modified size).

$$\begin{aligned} \prod PROD_{LAS} &= DATA * OU * OE * OI * DOCU \\ \prod PERS_{LAS} &= DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (5.11)$$

Note that in the product equation we include the DATA driver to measure the costs of the instantiation of the ontology, while the personnel equation incorporates exclusively factors related to domain experts.

If the shared ontologies need to be refined in order to fulfill local needs, the ontology users may decide between reusing existing ontologies, which have been developed by other users in the network, or by performing the desired modifications themselves. For the first case the effort multipliers are listed in equation 5.12. The second one is addressed by 5.13.

$$\begin{aligned} \prod PROD_{LAR} &= OU * OE * OM * OI * DOCU \\ \prod PERS_{LAR} &= DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (5.12)$$

Again the parameter $Size_{LAR}$ is understood as the total size of the reused ontologies. Integration costs arise, of course only in case external ontologies are reused (instead of separately modifying the shared ontology, the local adaptation may resort to existing modifications fulfilling the same requirements).

$$\begin{aligned} \prod PROD_{LAM} &= OM * OI * DOCU \\ \prod PERS_{LAM} &= DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (5.13)$$

5.4.1.3 The Costs of the Centralized Analysis and Revision Phase

$$\begin{aligned} PM_{CAR} = {} & Size_{CARR} * \prod PROD_{CARR} * \prod PERS_{CARR} * TOOL * SITE + \\ & Size_{CARM} * \prod PROD_{CARM} * \prod PERS_{CARM} * TOOL * SITE \end{aligned} \quad (5.14)$$

In this equation the first part computes the efforts needed to evaluate the changes performed locally, while the second part of the formula states for the efforts invested in executing these modifications. Note that $Size_{CARR}$ is the total size of the local ontologies. The parameter SITE accounts for eventual additional costs produced by the distributed setting. Again we elaborate the product and personnel multipliers (Eq. 5.15 and 5.16).

$$\begin{aligned} \prod PROD_{CARR} &= OU * OE * OI * DOCU * OE \\ \prod PERS_{CARR} &= OEXP * OCAP * LEXP * TEXP * PCON \end{aligned} \quad (5.15)$$

$$\begin{aligned} \prod PROD_{CARM} &= OM * OI * DOCU * OE * REUSE \\ \prod PERS_{CARM} &= OEXP * OCAP * LEXP * TEXP * PCON \end{aligned} \quad (5.16)$$

In the equations above the Ontology Evaluation (OE) multiplier appears repeatedly, since the board necessitates an evaluation of the submitted ontologies and a final evaluation of the new shared ontology.

5.4.1.4 The Costs of the Local Update Phase

$$\begin{aligned} PM_{LU} &= Size_{LUR} * \prod PROD_{LUR} * \prod PERS_{LUR} * TOOL + \\ & \quad Size_{LUI} * OI * DOCU * \prod PERS_{LUI} \end{aligned} \tag{5.17}$$

The parameter $Size_{LUR}$ designates the size of the "incoming" shared ontology, which is merged with the previous local one. For this reason, $Size_{LUI}$ is the sum of the two sizes involved in the merging process, the size of the new shared ontology plus the one of the existing local ontology.

$$\begin{aligned} \prod PROD_{LUR} &= OU * OI * OE * DOCU \\ \prod PERS_{LUR} &= = DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \tag{5.18}$$

$$\prod PERS_{LUI} = = DEXP * DECAP * LEXP * TEXP * PCON \tag{5.19}$$

Equations 5.5 to 5.18 offer the parametric setting necessary for estimating the person month effort invested in arbitrary DILIGENT processes. However, as aforementioned, the costs' dimension might be additionally used as a decision support factor on achieving an optimal distribution between centralized and local building phases during the ontology life cycle. In order to achieve this goal we need a reduced cost function – on the basis of the one elaborated in this section – which allows us to identify the most important dependencies between the major parameters of the process, as these dependencies are likely to be responsible for the realization of an optimal configuration of central and local building phases. This configuration is to be discovered in terms of the free parameters of the reduced formula.

5.4.2 The Reduced Cost Function

For the derivation of the reduced formula we start with the general DILIGENT cost equation 5.5 and consider the first level formulae for the corresponding person months calculations.

$$\begin{aligned} PM &= x_{new} * E_{new} + x_{reused} * E_{reused} + \\ &+ \sum_{i=1}^{n}(m_i * x_{las_i} * E_{las} + m_i * x_{lam_i} * E_{lam} + m_i * x_{lar_i} * E_{lar} + \\ &+ x_{carr_i} * E_{carr} + x_{carm_i} * E_{carm} + \\ &+ m_i * x_{lur_i} * E_{lur} + m_i * x_{lui_i} * E_{lui}) * p^i \end{aligned} \tag{5.20}$$

where the x_ks represent the sizes of the corresponding and the E_ks the multipliers. Note that the variation of the ontologies' size in each cycle is captured by the i indexation, while the variation of the cost driver values is modeled through the exponentially growing learning rate.

Let a be the average number of local changes submitted in every cycle $i = 1 \ldots n$, b be the average number of changes initiated by the board pro cycle and c the number of locally accepted changes. Further on let x_{la_i} be the size of a local ontology, which is submitted to the board after cycle i, x_{s_i} the size of the shared ontology obtained in the same cycle, and x_{lu_i} the size of the local ontology at the end of the cycle. The dependencies between the three sizes are as follows:

$$\begin{aligned} x_{la_i} &= x_{s_{i-1}} + a, \forall i = 1 \ldots n, x_{s_0} = x \\ x_{s_i} &= x_{s_{i-1}} + b, \forall i = 1 \ldots n, x_{s_0} = x \\ x_{lu_i} &= x_{s_i} + c, \forall i = 1 \ldots n \end{aligned} \tag{5.21}$$

If x is the size of the shared ontology in the first cycle ($x = x_{new} + x_{reused}$), then

$$\begin{aligned} x_{la_i} &= x + (i-1) * b + a, \forall i = 1 \ldots n, x_{s_0} = x \\ x_{s_i} &= x + i * b, \forall i = 1 \ldots n \\ x_{lu_i} &= x + i * b + c, \forall i = 1 \ldots n \end{aligned} \tag{5.22}$$

A simplification of the DILIGENTcost function is achieved if we reduce the effort multipliers related to the central board and to the user communities to single parameter with average values. That is, we abandon the difference between the effort multipliers involved in single activities of each process stage (e.g. the local analysis activity in the local adaptation phase), we obtain the formula below, in which E is again responsible for the centralized setting, while F represents average the local settings:

$$PM = x * E + M * F * \sum_{i=1}^{n}(x + (i-1) * b + M * a) * p^i +$$

$$
\begin{aligned}
&+ \quad M * F * \sum_{i=1}^{n}(x + i * b + c) * p^i + \\
&+ \quad E * \sum_{i=1}^{n}(M * (x + (i-1) * b + a) + b) * p^i
\end{aligned} \tag{5.23}
$$

In formula 5.23 we also assume a constant number of sites $m_i = M$ and an initial size of the ontology x.

If $n \rightarrow \infty$ then the formula 5.23 is further transformed to

$$
\begin{aligned}
PM \quad &\approx \quad x * E + M * F * (2x + M * a + c) * \frac{1}{1-p} + \\
&+ \quad M * F * b * \frac{p^2 + p}{(1-p)^2} + \\
&+ \quad E * (M * (x + a) + b) * \frac{1}{1-p} + \\
&+ \quad E * M * b * \frac{p^2}{(1-p)^2}
\end{aligned} \tag{5.24}
$$

5.4.3 Applications of the Reduced Cost Function

In order to come up with an approximation of the cost function corresponding to DILIGENT engineering processes we start with equation 5.24 and isolate the terms depending on the involved cost drivers E and F:

$$
\begin{aligned}
PM \quad &\approx \quad E * (x + \frac{1}{1-p} * (M * (x + a) + b) + M * \frac{p^2}{(1-p)^2} * b) + \\
&+ \quad F * M * ((2x + M * a + c) * \frac{1}{1-p} + b * \frac{p^2 + p}{(1-p)^2})
\end{aligned} \tag{5.25}
$$

The formula above is used to analytically describe alternative engineering scenarios in DILIGENT. We illustrate the usage of the cost function as an objective means for decision support for the following tasks:

- the identification of a specific engineering strategy: The DILIGENT methodology foresees a two-step engineering approach in which a first part of the shared ontology is jointly developed by domain experts and engineers, while the rest of the ontology evolves according to the needs of its users. Cost information might be useful to identify the sweet spot between the effort invested in centralized building and the remaining phases. A second engineering decision relates to the possibility

of taking into consideration external parties' ontologies (i.e. local versions of the shared ontology available at external sites) while performing modifications. A first possibility is to modify the shared ontology according to the local requirements and submit potentially locally relevant change requests independently of the requirements of other user communities across the network. In the second, reuse-oriented scenario the users first try to map their own requirements to local ontologies emerging across the network and to reuse these local ontologies instead of introducing new change requests. The decision on one of the alternatives could be documented by means of cost information.

- the identification of the optimal meeting frequency: For an optimal process execution one needs decision criteria to estimate the frequency of the board meetings and implicitly a rate for the amount of submitted and approved changes to the shared ontology. Since every new board meeting is related to (basic) costs for the centralized analysis and the local updates too frequent meetings are expected to produce an overload both on the side of the ontology engineers and of the users.

In order to analytically describe the aforementioned scenarios, we consider a simplified version of the DILIGENT process, in which the costs of the local updates are negligible. In this case, equation 5.25 is transformed to

$$\begin{aligned} PM &\approx E*(x+\frac{1}{1-p}*(M*(x+a)+b)+M*\frac{p^2}{(1-p)^2}*b)+ \\ &+ F*M*((x+M*a)*\frac{1}{1-p}+b*\frac{p^2}{(1-p)^2}) \end{aligned} \tag{5.26}$$

5.4.3.1 1. Scenario: The Size of the Initial Ontology

In order to analyze the impact of the initial size of the ontology on the overall costs required to create an ontology of a given final size we compare the costs defined in equation 5.26 with the ones implied by an initial ontology of size $x+\alpha$. In this situation, we assume that the number of changes required by the users decrease with the size of the initial ontology with a parameter β. A large initial ontology is profitable if

$$E*M*(\alpha-\beta)+F*M*(\alpha-M*\beta)-\alpha*F*M>0 \tag{5.27}$$

Inequality 5.27 is equivalent to $E*(\alpha-\beta)>F*M*\beta$, which means that increasing the size of the ontology is profitable as long as the costs arisen by this activity for analysis and evaluation do not overcome the costs required to originally build the additional concepts. The last inequality will be not fulfilled for a sufficiently high number of sites M, which implies that we need to start with a small shared ontology in case we need to handle

a high number of sites. In case the number of sites is sufficiently low, the satisfiability of the inequality is influenced by the ratio between the productivity of the board and of the ontology users.

5.4.3.2 2. Scenario: Reuse-oriented vs Isolated Building

In order to detect the impact of local ontology reuse on the overall costs, we proceed in a similar manner as in the first scenario by comparing the difference arising from modifying a number of α concepts instead of trying to reuse them from external sources. By replacing the number of changes a in 5.26 with $a + \alpha$ we obtain that reusing existing conceptualizations is profitable only if $E * M * \alpha + F * M * \alpha > F * M^2 * \alpha$. Again, reuse is more feasible for scenarios with a relatively low number of sites. If M is sufficiently high, the inequality above is not satisfied anymore.

5.4.3.3 3. Scenario: Frequency of Board Meetings

In order to analyze the optimal frequency of board meetings, which influence the number of submitted and accepted changes (the frequenter the meetings are, the less changes are submitted pro cycle). If α is the difference between the number of submitted changes for a higher number of development cycles, β is the difference at the level of approved changes – the costs for local updates are ignored – then the difference implied by these three parameters w.r.t. the person months efforts in two consecutive cycles is determined from equation 5.23 as:

$$\begin{aligned} PM_i * P - PM_{i+1} &= M * F * ((i-1) * \beta + M * \alpha - b_{i+1} + \\ &+ E * (M * (i-1) * \beta - M * b_{i+1} + M * \alpha + \beta) \end{aligned} \quad (5.28)$$

In this case $PM_i * P - PM_{i+1} > 0$ if $(i-1) * \beta + \alpha > b_{i+1}$ and $M * \alpha + (i-1) * \beta > b_{i+1}$. The latter is satisfied for a sufficiently high number of sites M, while the former depends on the parameter i. If $\beta + \alpha > b_{i+1}$ increasing the number of meetings is feasible independent on the number of sites or on the learning rate.

5.5 Data Collection and Model Calibration

In this report we demonstrated the ways the generic cost estimation model ONTOCOM was applied to the ontology engineering methodology DILIGENT. The alignment of the model to this particular methodology also revealed the limitations of the model w.r.t. a complete coverage of ontology engineering aspects. As a consequence the estimation model was refined with cost drivers such as ”Ontology Integration”. In the same time,

the alignment can be seen as a significant step towards the final validation of ONTOCOM, which is performed according to the quality framework described in Section 5.2. However, the usability of the model in real-world settings is primarily dependent on the accuracy of its results, achieved after calibrating the a-priori parameter values on the basis of historical project data. For this purpose, we analyzed the technical means which can be used for the calibration and produced an online questionnaire for the collection of the data. These two issues are described in the remaining sections.

5.5.1 Technical Realization of the Data Collection

For the realization of an online tool for data collection we made use of the Open Source survey software PhpESP (available at `http://sourceforge.net/projects/phpesp/`), which offers basic functionality for the generation of public surveys. The data collection procedures is foreseen as a set of questions by which the user is required to provide introductory information about a specific project (i.e. an ontology) and to specify the values of the parameters included in the cost model. As a result of the survey, the data is stored in a relational database and is exported to a statistical component in order to be used for the calibration of the model.

Figure 5.1 depicts the introductory section of the data collection survey, while Figure 5.2 shows an excerpt related to the cost drivers ”DATA” and ”DCPLX”. For each cost driver[PBM05a] we provide a short explanation of the scope and associated decision criteria, which are intended to be used to aid the data provider in specifying the rating value of the driver. Finally we can export the collected data as shown in figure 5.3 and calibrate the model.

A current version of the survey is available online at `http://kompass.mi.fu-berlin.de/phpESP/public/survey.php?name=ontocom2final_260905`.

5.5.2 Calibration Method

The data collected in the survey is used to calibrate the ONTOCOM cost estimation model. Before we explain the methods used to calibrate the model, we need to emphasize the restrictions of any calibration. (1) Due to the number of cost drivers we need a high number of observations to calibrate the model in a statistically significant way. Any calibration with less than approximately 300 data points will not be significant from a statistical point of view. Thus any calibration can only be seen as an indication of the direction. (2) Experience in other fields with cost estimation models suggests, that a calibration for a particular company or project team yields more accurate estimations than a general purpose calibration. Our calibration can therefore only serve as an example for the calibration process, rather than an accurate model calibration. Nevertheless, the calibration is useful, as project teams can compare their estimations against a general average

as provided by us. (3) A calibration uses historical data to estimate future outcomes. Although the average and the variation observed in the historical data may also be observed

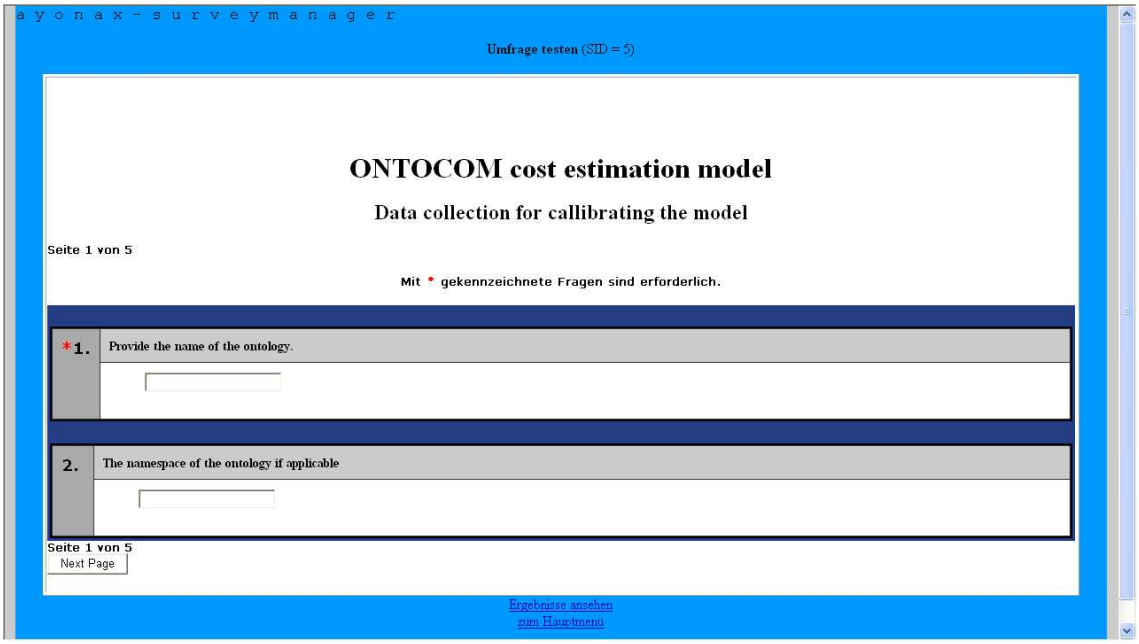

Figure 5.1: ONTOCOM data collection: introductory questions

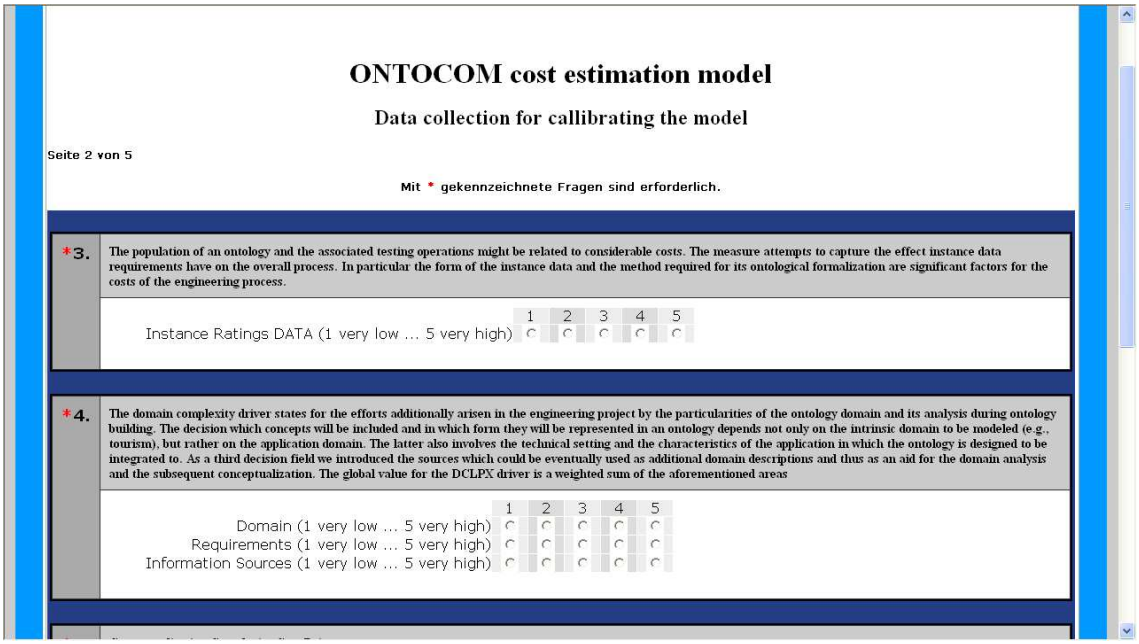

Figure 5.2: ONTOCOM data collection: cost drivers

Parameter	Description
A	adjustment parameter
$Size$	The size of the ontology
$DCPLX(CD_{X1})$	Effort multiplier for domain complexity
$OE(CD_{X2})$	Effort multiplier for final ontology evaluation complexity
$REUSESize$	Size of the reused ontology
OU	Reused ontology understandability

Table 5.13: Simplified cost model factors

in future projects, any specific project can still require significantly more or less effort to build the ontology than the predicted one.

The number of cost drivers defined in ONTOCOM is too high, to work through a conclusive example. In order to explain the calibration method to refine the ONTOCOM cost model we introduce a very simple cost model. For the simplified cost model we provide a complete example.

Our simplified cost model consist of six factors as listed in table 5.13.

$$AdSize_X = Size_X - (1 - \%reuse) * REUSESize * OU \tag{5.29}$$

Figure 5.3: Data export from phpESP

Rating	DCPLX			OE			OU		
	E1	E2	Av.	E1	E2	Av.	E1	E2	Av.
very low	0,6	0,8	0,7	0,6	0,8	0,7	0,6	0,8	0,7
low	0,7	0,9	0,8	0,7	0,9	0,8	0,7	0,9	0,8
nominal	1	1	1	1	1	1	1	1	1
high	1,1	1,3	1,2	1,1	1,3	1,2	1,1	1,3	1,2
very high	1,8	2,0	1,9	1,8	2,0	1,9	1,8	2,0	1,9

Table 5.14: Delphi result

$$PM_X = A * AdSize_X * \prod_{i=1}^{2} CD_{Xi} \tag{5.30}$$

For our simplified cost model the Delphi method resulted in the following expert estimations for our effort multipliers (in table 5.14, the estimation of the experts are abbreviated with $E1$ and $E2$, respectively. Average values are termed by $AV.$).

We collected data from six ontology building projects. The results are summarized in table 5.15 ($RSIZE$ is the size of the reused ontologies, while the DCPLX columns correspond to the three decision components defined for the cost driver Domain Analysis Complexity, as introduced in Section 5.3).

				DCPLX					
Ontology	SIZE	PM	% newly build	req.	con.	info.	RSIZE	OU	OE
swpatho1	1300	5	20	5	1	4	1040	2	5
opjk	700	2,6	100	5	4	5			5
ArguOnto	200	2	100	4	2	4			2
COS	75	2,5	100	5	3	3			5
OMV	300	2,5	100	3	2	4			3
VDO	1400	0,5	100	4	4	4			1

Number	Rating
1	very low (VL)
2	low (L)
3	nominal (N)
4	high (H)
5	very high (VH)

Table 5.15: Data collection

The collected data is then adjusted in order to apply the calibration. In this step the ratings for the domain complexity are averaged and the size of the reused ontology and the final size are combined 5.16.

The data collected from real projects can now be used to calibrate our model. Linear regression is the adequate method to find the adjusted parameters. We reformulate equation 5.30 in order to apply later on linear regression and introduce a parameter β_i as an exponent for the cost drivers. β_i is a scaling factor, by which the existing parameters should be scaled in order to fit the model. We recall that α is factor to represent a learning rate, in other case also used to model economies of scale.

$$PM_X = A * AdSize_X^{\alpha} * \prod_{i=1}^{2} CD_{Xi}^{\beta_i} \tag{5.31}$$

We apply the logarithm to equation 5.31 and can now apply a classical linear regression to our data to estimate β_i. This step is only possible if our data is distributed exponentially, thus we have significantly more data points with a low number of entities than with a high number of entities. We omit this test for our example, but will do so for the final calibration.

$$\ln(PM_X) = \ln(A) + \alpha * \ln(AdSize_X) + \sum_{i=1}^{2} \beta_i * ln(CD_{Xi}) \tag{5.32}$$

The resulting matrix of data points can be used to calculate the covariance matrix and the correlation matrix. In particular the correlation matrix is helpful, to identify cost drivers which are highly correlated and can thus be integrated into one. For the sack of completeness we have listed the results of the correlation analysis in table 5.17. The analysis of the correlation matrix reveals that for our limited data set the total effort for building the ontology is highly correlated with the extend of the ontology evaluation activity. Furthermore domain complexity and ontology evaluation are correlated. Surprisingly, the size of the ontology and the required effort to build it are inversely correlated, which implies the larger the ontology becomes the less effort one has to spend building it. This result shows, that an estimation model, calibrated only based on historical data could in fact be misleading. We survey several methods to overcome this problem later.

Ontology	AdSize	PM	DCPLX	OE
swpatho1	635	5	3	5
opjk	700	2,6	5	5
ArguOnto	200	2	3	2
COS	75	2,5	4	5
OMV	300	2,5	3	3
VDO	1400	0,5	4	1

Table 5.16: Adjusted collected data

	AdSize	DCPLX	OE	PM
AdSize	1,00			
DCPLX	0,27	1,00		
OE	-0,25	0,41	1,00	
PM	-0,40	-0,09	0,74	1,00

Table 5.17: Correlation matrix for our example

	A	DCPLX	OE	α
m_i	0,84	-1,37	1,48	-0,03
$s(m_i)$	1,79	1,34	0,70	0,30

Table 5.18: Results of the linear regression

In table 5.18 we have summarized the results of the linear regression. Applying the results to the parameters we get new parameters according to table 5.19 ($Av.$ means "Average").

We can now compare the predicted effort according to our model before and after its calibration. As recognized before some of the results of the linear regression are counter intuitive. Different options exist to overcome this problem.

5.5.2.1 Linear Combination

The expert ratings found in the Delphi experiment are a-priori estimations for our parameters and incorporate knowledge about the underlying activities. Using only historical data to calibrate the model would thus waste this knowledge. A natural solution is to combine the values estimated by the experts with the parameters found from the historical data. In the literature a combination which weights the expert values with 90% and the values from the linear regression with 10% is proposed.

Rating	DCPLX			OE			A			α		
	Delphi	Data	Av.	Delphi	Data	Av.	Delphi	Data	Av.	Delphi	Data	Av.
VL	0,7	1,63	0	0,7	0,59	0	1	2,31		0,98	-0,03	
L	0,8	1,36	0	0,8	0,72	0	1	2,31		0,98	-0,03	
N	1	1	1	1	1	1	1	2,31		0,98	-0,03	
H	1,2	0,78	0	1,2	1,31	0	1	2,31		0,98	-0,03	
VH	1,9	0,41	0	1,9	2,59	0	1	2,31		0,98	-0,03	

Table 5.19: Parameter estimation from experts and based on the data

5.5.2.2 Bayesian Linear Models

The linear combination of expert estimations and historical data is not optimal. The combination should take into account the number of data points used for the linear regression and the variance observed in the expert rating as well as in the data points. A factor which all experts have given the same rating, while the linear regression results in a high variance should be influenced less by the data than by the experts. Bayesian analysis is a way to achieve the desired outcome. [DC99] provides an exhaustive explanation of the application of Bayesian analysis for cost estimation models. As Bayesian analysis requires methods which go beyond the standard statistical functions offered by eg. Microsoft Excel software packages such as produced in the Bayesian inference Using Gibbs Sampling (BUGS) project `http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml` must be used. An Excel package `http://www.jstatsoft.org/v14/i05/v14i05.pdf` for the BUGS software exists[6].

Later on we consider to offer an online service which calibrates the model automatically when new information is available. In this case a PHP based service might be useful[7].

For our purposes we can weight the parameters based on the variance of the linear regression parameters m_i. In this case the weight of parameter is calculated as in equation 5.33.

$$CD_{Xi}^{new} = CD_{Xi}^{expert} * (1 - \frac{1}{s(m_i)^2 + 1}) + CD_{Xi}^{data} * \frac{1}{s(m_i)^2 + 1} \tag{5.33}$$

As the Bayesian analysis is a very sophisticated method, our data however is still very limited we opt for the linear combination of expert and estimated parameters. In table 5.20 we compare the accuracy of our estimations w.r.t. different parameter settings. As accuracy we define the percentage of estimations, which lie within a certain range of the actuals.

Alternative calculation

As we have already observed from the correlation matrix, size and effort are invers correlated. This has unintended effects on the estimated learning rate. In order to overcome this problem from the beginning, we can assume a learning rate of 1, thus we do not assume any learning. This changes eq. 5.31 slightly and thus 5.32 as shown in eq. 5.34

[6] Further examples can be found at `http://www.biostat.umn.edu/~sudiptob/pubh5485/BayesianLinearModelText.pdf`

[7] An explanation of the PHP based implementation of Baysian networks is found here `http://www.devshed.com/c/a/PHP/Implement-Bayesian-inference-using-PHP-Part-1/`

Name of the Ontology	PM actual	Expert		Linear 90% expert		Linear 20% expert		Linear 0% expert		Simple Bayesian	
		Accuracy									
		30%	50%	30%	50%	30%	50%	30%	50%	30%	50%
		0,17	0,17	0,0	0,67	0,33	1,0	0,17	1,0	0,33	0,83
		Estimations									
swpatho1	5	1,2		3,6		4,5		4,8		4,0	
opjk	2,6	2,5		6,4		3,2		2		3,7	
ArguOnto	2	0,2		1,5		1,4		1,4		1,5	
COS	2,5	0,2		4,5		4,2		4		4,0	
OMV	2,5	0,3		1,9		1,9		1,9		1,9	
VDO	0,5	1,2		1,4		1,0		0,8		1,1	

Table 5.20: Effort estimation based on expert estimation and historical data

	A	DCPLX	OE
m_i	-5,20	-3,29	2,48
$s(m_i)$	0,68	2,60	1,37

Table 5.21: Results of the linear regression - alternative

$$\ln(\frac{PM_X}{AdSize_X}) = \ln(A) + \sum_{i=1}^{2} \beta_i * ln(CD_{Xi}) \tag{5.34}$$

In this case the results of the linear regression are as shown in table 5.21

Future path

We have demonstrated with a simplified example the process of calibration for the ONTOCOM model. As we are now gathering more data of real world ontology building efforts we will soon be in a position to calibrate the complete model. The calibration will probably result in an adaption of the cost drivers. Some cost drivers might be highly correlated and can thus be joined. Others might have such a big impact on the final estimation that they can be divided into more than one cost driver. As soon as more accurate date is available we will continue and report the results of the calibration.

5.6 Conclusions

In the last couple of years we witness a change of focus in the area of ontologies and ontology-based information systems: while the application of ontologies was restricted

for a long time to academia projects, in the last ten years ontologies have become increasingly relevant for commercial applications as well. A first prerequisite for the successful introduction of ontologies in the latter setting is the availability of proved and tested Ontology Engineering methodologies, which break down the complexity of typical engineering processes and offer guidelines to monitor it. Existing methodologies have proven to fulfill these requirements. A further prerequisite is, however, the availability of cost information for the ontology building effort so that the project initiator can compare the estimated costs against the prospected utility of the ontology. Research in this field is not very advanced yet, but ONTOCOM is an initial attempt in this direction.

In this chapter we have described the ONTOCOM cost estimation model. ONTOCOM is a parametric cost estimation model, which assumes a linear relation between the size of an ontology and a serie of cost drivers which are determined according to the project setting. The model can be applied in the early project phases (such as the feasibility study) in order to compute an estimation of the effort (expressed in person months) arisen by building, reusing or maintaining ontologies.

We have shown that the cost model ONTOCOM can be aligned to a specific ontology engineering process such as DILIGENT, which covers all major phases of ontology engineering, such as building, reuse and maintenance. As a result ONTOCOM incorporates now 25 cost drivers which cover efforts for ontology building, reuse and maintenance, divided into three categories: product, personnel and project cost drivers.

The cost function specific for DILIGENT process was defined according to the ONTOCOM model and the alignment of the cost drivers to the activities they have an impact on. The cost function was then simplified in order to enable the usage of cost information as decision support for three engineering scenarios, which were specified during this work: 1.) finding the optimal size of the initial ontology, 2). the extension of reuse at the local sites, and 3). the optimal frequency of board meetings. In summary, our analytical investigations on the basis of the simplified cost function revealed that the decisions should depend on the number of sites the ontology is used at, and the capabilities of the ontology engineers and its users respectively.

In order to calibrate the ONTOCOM model, thus to find the right parameters for the different cost factors, we set-up an online survey to collect data from existing ontology engineering projects. So far the survey was utilized to capture data from 28 projects. To minimize misinterpretations of the cost drivers and their ratings, the authors of this report interviewed members of the corresponding engineering teams and entered the data themselves. Up-to-now the data collection covers historical projects at our institutes (Free University of Berlin, University of Karlsruhe) and the EU project SEKT. Data collection from the Knowledge Web project and other organizations will follow.

Although the number of collected data points is still insignificant, we can already draw some preliminary conclusions and point to future research issues. The interviewed persons could describe their experiences in the ontology building effort with the proposed cost drivers, which demonstrates the usability of the cost model for the intended class of

engineering projects. The alignment to the elaborated DILIGENT methodology definitely contributed to a large extent to this satisfactory coverage. Nevertheless future alignments with other ontology engineering methodologies are expected to help us complete and refine ONTOCOM's list of cost drivers.
Further on our experiences in applying ONTOCOM so far suggest that the complexity of ontology evaluation has a significant impact on the overall project costs. Consequently, this indicates that better support for ontology evaluation could yield important benefits. Another potential cost-relevant parameter appears to be the number of domain experts from different domains, building a single shared ontology. The extension of the model with this coordinate, which is not supported by the current version, is subject of future investigations.

Beyond cost estimation, the list of cost drivers was found helpful for ontology engineers to breakdown activities related to a specific ontology building process during the feasibility study.

Chapter 6

DILIGENT ontology learning process

Note: Work in progress The descriptions, though, are not yet elaborated, as we have to gather the results of our case studies first. Nevertheless, did the presentation and tutorial given to the Legal case study partners, already facilitate their effort to learn ontologies from legal documents.

6.1 Motivation

In section 4.2 we mentioned that the automatic learning of ontologies is an integral part of the DILIGENT methodology. The integration, however, was so far coarse and the application in the case study required a more fine grained process description. Therefore we have conceived an ontology learning process model, which extends the existing process description. The process model is currently applied and evaluated in the legal case study. So far we can only report on the initiale set-up of the process model and the roles, activities, decisions and stages defined.

6.2 Process

We propose methods to (semi-)automatically generate ontologies from other existing information sources. The process can be applied in a standalone manner or in conjunction with manual ontology building. The result (the learned ontology) should be evaluated by domain experts. The process defining roles, stages, activities, decisions, input- and output values and tools is depicted in fig. 6.1.

We define the following eight process stages:

1. Feasibility study

2. Requirements specification
3. Selection of information sources
4. Selection of ontology learning tools
5. Learning preparation
6. Learning execution
7. Ontology evaluation
8. Ontology integration

The user can go fourth and backward between the process stages depending on the outcomes of a respective process stage. Although the different activities roles and decisions are not yet elaborated, we decided to present our initial results here. In upcoming deliverables we will provide more detailed descriptions. Besides we introduce a small example, as a work through for the proposed process. In the example we assume that the ontology engineers should build an ontology for the travel domain.

6.2.1 Feasibility Study

Roles Domain experts, ontology engineers

Input factors Ontology requirements specification document (ORSD)

Example In our ontology learning example from the travel domain the ORSD is defined as in table 6.1.

Output factors Risk analysis document analyzing Potential risk factors and proposing a Risk management *Example*

- Texts contain very detailed information
- Partially telegraphic writing style
- Unsure about overlap of activities
- Texts written by different authors

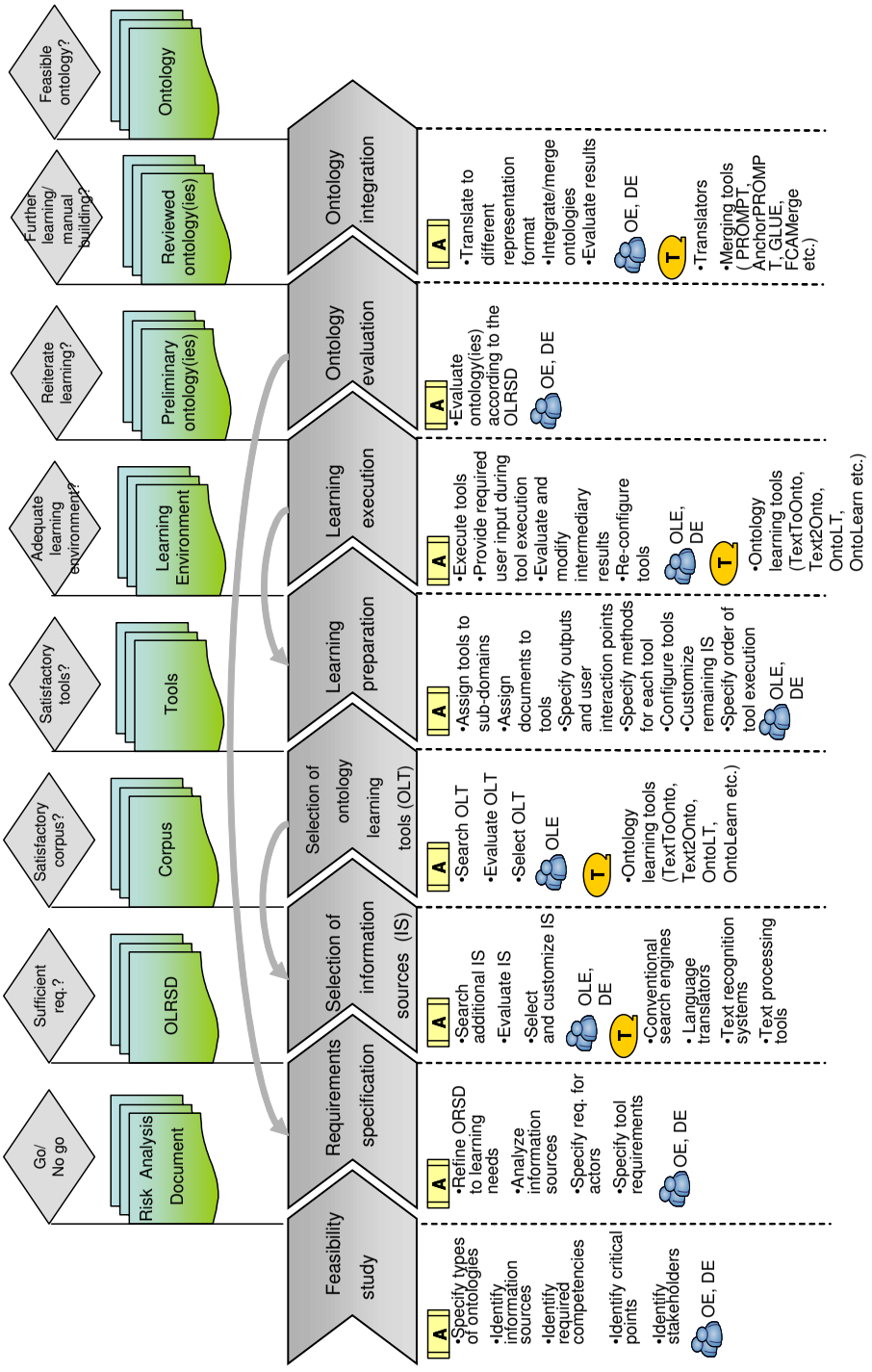

Figure 6.1: Stages, Roles and Activities in ontology engineering with ontology learning

Activities

- Specify types of ontologies to be learned. This should be done considering the sub-domains from the ORSD.
  - Domain ontology
  - Application ontology
  - Top-level ontology

  Example European tourist information (Domain ontology)

- Identify information sources useful for the learning process *Example* Lonely planet documents on my hard disk

- Identify required competencies
  - Linguistic expertise
  - Domain expertise
  - Tool know-how
  - Language know-how

  Example TexttoOnto (Tool know-how) and English (Language know-how)

- Identify critical points in the learning process
  Example Too detailed information in the text documents, free text, large set of documents used in learning might imply evaluation overload

- Identify stakeholders of the learning process
  Example Web portal owner

Property	**Value**
Goal, Domain, Scope	Tourist information about activities, attractions, environment etc. in European cities
Design Guidelines *e.g.* Number of concepts, level of granularity, terminology	200 concepts, only major activities etc. with at least one occurrence, concept labels in English and Spanish
Knowledge Sources *e.g.* Domain experts, data bases, text etc.	Lonely planet Web site
Potential Users, Usage Szenarios	Tourists, Selection of travel destination
Supported Applications	Web portal
Usage of competency questions	What can one do in Barcelona? Where can I sky and bike? Which city with good weather has a museum exhibiting the art of Picasso?

Table 6.1: ORSD for the travel domain

Decisions It should be decided if to proceed ontology building by learning or not.

Example Yes

Tools None

6.2.2 Requirements Specification

Roles Domain experts, ontology engineers

Input factors Ontology requirements specification document (ORSD), risk analysis document

Output factors Ontology learning requirements specification document (OLRSD)

- Requirements regarding ontologies (ORSD)
- Requirements regarding ontology learning tools
- Requirements regarding actors

Activities

- Refine ORSD and match it to the requirements of ontology learning
  Example Activities (Domain ontology), Environment (Domain ontology), Facts (Domain ontology), Attractions (Domain ontology), Other (Domain ontology)
- Analyze information sources
  Example Lonely planet is one information source. Texts for different sub-domains should be separated
- Specify tool requirements, w.r.t. Inputs, outputs, language, type of learning method etc.
  Example Files in text format, concepts, relations, hierarchy, English, analysis of texts
- Specify requirements regarding actors
  - Level of expertise for domain experts and ontology engineers etc.
  - Need for additional experts

  Example Ontology learning expert required, KAON expert required

Decisions Sufficient requirements (to perform learning)

Tools None

6.2.3 Selection of Information Sources

Roles Domain experts, ontology learning experts

Input factors OLRSD, information sources

Output factors Selected information sources (corpus)

Activities

- Search additional information sources
  Example Tourism corpus for the same domain in Spanish
- Evaluate information sources (according to the OLRSD), w.r.t. Language, domain, representation, structure etc.
  Example Heterogeneous Spanish corpus
- Select and customize information sources. This includes their translation, digitalization or other form of processing
  Example Copy text according to domain into different directories. Spanish must be transformed to a uniform representation format

Decisions Information sources feasibility (w.r.t. OLRSD)

Example English corpus is feasible for learning. Spanish corpus requires considerable customization.

Tools Conventional search engines, language translators, text recognition systems, text processing tools

6.2.4 Selection of Ontology Learning Tools

Roles Ontology learning experts

Input factors OLRSD, information sources

Output factors Selected ontology learning tools

Activities

- Search Ontology Learning Tools (OLT)
  Example In our case TextToOnto, Text2Onto
- Evaluate OLT (according to the OLRSD)
  - User friendliness (interface, transparency, interaction points etc.)
  - Applied linguistic methods
    Example POS tagger, TFIDF, see D3.3.1
  - Input and output characteristics
    Example English and Spanish ASCI texts, KAON ontologies
  - Additional required knowledge sources etc.
    Example GATE
- Select OLT

Decisions Tool feasibility (w.r.t. OLRSD)
Example Most of the requirements can be met by TextToOnto *cf.* 6.2.

Requirement	**TextToOnto**
Files in text format	OK
Concepts	OK
Relations	OK
Hierarchy	OK
English, Spanish	OK
Analysis of texts	OK
Instances	No

Table 6.2: Evaluation of tool feasibility

Tools Ontology learning tools, such as TextToOnto, Text2Onto, OntoLT, OntoLearn

6.2.5 Learning Preparation

Roles Domain experts, ontology learning experts

Input factors OLRSD, information sources, ontology learning tools

Output factors Configured learning environment

Activities

- Assign tools to sub-domains
  Example For all five domains use TextToOnto
- Assign documents to tools
  Example English documents corresponding to the five categories with TextToOnto English; Spanish documents with TextToOnto Spanish
- Specify outputs
  Example OWL ontology
- Specify user interaction points
  Example (1) Learn concepts, customize concepts; (2) Learn hierarchy, customize hierarchy; (3) Learn relations, customize relations
- Specify methods for each tool
- Configure tools
  Example Select appropriate filters according to text corpus
- Customize remaining information sources
- Specify order of tool execution
  Example Parallel execution

Decisions Feasible learning environment

Tools None

6.2.6 Learning Execution

Roles Domain experts, ontology learning experts

Input factors OLRSD, information sources, ontology learning tools

Output factors Preliminary ontology(ies)

Activities

- Execute tools
- Provide required user input during tool execution
- Evaluate and modify intermediary results
- Re-configure tools

Decisions Either the learning process is Completed, or a reiteration of the learning process with revised parameters must be performed.
Example English ontologies learned; Spanish ontology requires new parameter configuration

Tools Ontology learning tools, such as TextToOnto, Text2Onto, OntoLT, OntoLearn

6.2.7 Ontology Evaluation

Roles Domain experts, ontology engineers

Input factors OLRSD, preliminary ontology(ies)
Example English ontologies, Spanish ontology

Output factors Reviewed ontology(ies)
Example English ontologies

Activities Evaluate ontology(ies) according to the OLRSD
Example English ontologies satisfy the requirements, class instance distinctions should be corrected manually
Spanish ontology unfeasible, translate English terminology to Spanish

Decisions

- Feasible learning results
- Reiteration points
  Example No
- Proceed with other forms of ontology building
  Example Manual modifications

Tools Currently none, but as described we are currently developing tools to facilitate ontology evaluation.

6.2.8 Ontology Integration

Roles Domain experts, ontology engineers

Input factors ORSD, learned ontologies
Example Five ontologies in English

Output factors (Preliminary) ontology
Example Tourism ontology

Activities

- Translate to different representation format
  Example Translate KAON to OWL
- Integrate/merge ontologies
  Example In our case necessary
- Evaluate results
  Example Insert upper level concepts

Decisions Feasible ontology
Example Yes

Tools Translators, merging tools such as PROMPT, AnchorPROMPT, GLUE, FCAMerge etc.
Example Manual integration process

6.2.9 Future Work

The ontology learning process model will be evaluate and refined in the course of the project. We will elaborate on the structure of risk analysis document and OLRSD. The decision decision criteria for the enumerated process steps will be specified. For existing ontology learning tools we will provide a detailed classification. From the case studies we will compile best practices for learning execution.

Chapter 7

Evaluation of the DILIGENT methodology

Evaluation is a process to compare different approaches to solve a certain problem [Hou80].

An evaluation is often associated with a decision to be made but this is not a necessity. Different approaches exist to evaluate procedures. The indented audience and their requirements entail the aspects which distinguish them. The intended audience of an evaluation ranges from economist and managers requiring numbers and facts to choose the most efficient procedure to practitioners being more interested in experiences in the application of the procedure.

In order to obtain the required output information the evaluation approaches follow different methodologies. From an epistemological point of view there are objectivist and subjectivist approaches to evaluation. Evaluation approaches following a objectivist epistemology concentrate on observable facts, quantitative techniques, strict procedures and reproducible results. Examples of these approaches to evaluation are the "Systems analysis" and the "Goal free" model. The approaches following a subjectivist epistemology observe subjective impressions. Informal interviews, personal judgement and experiences in the case study are the means to collect multiple perspectives on the evaluated procedures. Examples of these approaches to evaluation are the "Professional review" and "Case study" model. In any case for the evaluation to be of value the audience must trust the procedures of the evaluator. The evaluation should thus be "true, credible, and right" [Hou80, p. 250].

The way the evaluator claim credibility differs depending on the approach. The evaluator should be unbiased, and he should be interested in the findings but not favor any of the approaches. In case of the objectivists approach to evaluation the audience should agree with the facts, while in the subjectivist case the audience should agree with the experiences made in the case studies.

An evaluation should always start with the definition of a testable hypothesis. As the primary method of data collection is the observation of the applied procedure under evaluation, a key criterion for an evaluation to be credible is the possibility to replicate it. All evaluation procedures should thus be externalized and explicit.

For example the evaluation of an procedure by means of a "Case study should report on the important experiences in it, the credentials of the participants, e.g. of the professional reviewers or the participants in the case study and communicate important insights, which are not standardized upfront. However, insights might vary considerably between two case studies since people change.

It is not possible to select "the" best evaluation method, since they all have different objectives, focus on varying aspects and draw diverse conclusions. Each approach to evaluation has strength and weaknesses. Following only one evaluation approach might thus be misleading and capture all aspects of the evaluated approach. Regarding the evaluation of ontology engineering methodologies[1] no clear "winner" could yet be established. Therefore we have decided to evaluate the methodology according more than one model. We have opt for the "Goal free", "Professional review" and finally the "Case study" model, because the disadvantages of one model are covered by the advantages of the other models so that they complement each other. Other methodologies have been evaluated according to at most two of these models[2]. In the next section we describe the three models, explaining the assumptions underlying each model and the process of evaluation, defining the inputs and outputs and illustrate the advantages and disadvantages of the evaluation approaches. In section 7.2 we compare the DILIGENT methodology to major ontology engineering methodologies available in the literature. The results of the "Professional review" are mentioned throughout the deliverable and have already lead to adaptations of the methodology. In section 7.4 we report the results of a first application of the DILIGENT process in a case study. For further experiences in and evaluations of the methodology in case studies we refer the interested reader to deliverable 7.2.1.

7.1 Selected Methods for Evaluation

In this section we describe with more detail the selected evaluation models, namely the "Goal free", "Professional review" and finally the "Case study" model. We provide a short description of the model and of the process to conduct the evaluation. Each model takes different inputs and can deliver respective outputs. The models have certain advantages and disadvantages.

[1] Note that at this point we do not aim at evaluating the resulting ontologies but the process itself.

[2] *e.g.* The OTK methodology and Methontology have been evaluated according to the case study and goal free model.

7.1.1 Goal Free

Description The "Goal free" approach to evaluation takes its name from its purpose. The evaluation is performed for no particular reason, but to compare different procedures according to a common set of criteria. It is then up to the reader of the evaluation to assign personal preferences to the single criteria and select the procedure which fits the reader best. The criteria can be qualitative as well as quantitative. In computer science the evaluation of algorithmic methods according quantitative measures prevails. For methodologies, as in our case, it is difficult to define quantitative criteria. The quality and efficiency of the ontology engineering effort could be measured taking into account the quality, size and total effort spend to build the ontology. It is, however, difficult to compare these results for one methodology with the results for another methodology as the effort spend depends not only on the used methodology but on many other factors, such as ability of the team, experience of the team, project environment which are not easily eliminated. Therefore, our evaluation takes into account only qualitative measures which can be unbiasedly evaluated.

Process The goal free evaluation starts with the selection of relevant evaluation criteria. The evaluator should take into account all aspects which a later reader of the evaluation might find interesting. The evaluation criteria should be clearly defined, they should not be overlapping, and general enough to cover all evaluated methods. Furthermore the criteria should be relevant for the intended application of the procedures. For each evaluation criteria the evaluator assigns a value to the evaluated procedures.

Input The basis for a goal free evaluation is a objective set of evaluation criteria for the methods under evaluation.

Output The goal free evaluation provides the reader with an objective analysis of a set of criteria for a number of methods.

Advantages The evaluation method is objective, as all models must adhere to the same evaluation criteria. The evaluation can be used for different purposes, as the criteria are not weighted. The identification of evaluation criteria, provides the reader of the evaluation with a quick overview of the relevant issues for the particular procedures.

Disadvantages The identification of evaluation criteria is crucial for this evaluation model. The evaluation criteria should be selected in a way, that particular advantages of the evaluated models are comparable. If the evaluator is not completely familiar with the evaluated models, wrong judgement for a criteria might be the result.

7.1.2 Professional Review

Description The professional review model relies for evaluation purposes on the experiences of knowledge people in the area of interest. A number of professionals review a procedure w.r.t. its plausibility and base their judgement and recommendations on their own experiences and their own knowledge. They can use predefined evaluation criteria to structure the assessment.

Process The professionals carefully examine the process description. If evaluation criteria are available, they judge the evaluated process according to those criteria. They make recommendations for changes in the process if necessary.

Input Evaluation criteria can be an input to this model.

Output This evaluation model produces expert opinions regarding the plausibility and understandability of the evaluated method.

Advantages The professional review model reduces bias, since all aspects of a procedure are evaluated. Hidden consequences of the evaluated procedure can be detected, because the evaluator brings all his knowledge and experience in the evaluation. The professional review model is typically applied after the first creation of the model in order to eliminate obvious errors and to get a different perspective on the evaluated procedure.

Disadvantages The outcome of this evaluation depends on the capabilities and opinion of the evaluator. The evaluation depends on the availability of experts and can not regularly be repeated.

7.1.3 Case Study

Description The main objective of the case study model to evaluation is to understand the process under evaluation. The target audience follows the process and the evaluator tries to capture as many information as possible from the execution of the process.

Process We assume that the target audience knows the process to evaluate. The target audience accomplishes their tasks according to the new process model, or – if the process is an addition to the participants regular tasks – adds new tasks to their daily work. A case study can have a predefined duration. The evaluators start the case study analysis with a predefined research question (*cf.* [YC03]). They should consult additional information sources and perform a literature review in order to formulate a precise research question.

They determine the data gathering and analysis techniques for the case study; interviews, surveys and observation are valid data gathering techniques, which all require a specific process so that the validity of the observations is ensured. In the course of the case study the evaluator collects the data from the different participants in the process. Many participants should provide data, in order to get varying views on the process. The analysis of the data should expose the important issues and relevant findings. In the aftermath of the case study the evaluator describes the result, taking into account the organizational setting, the participants situation, the participants reports and his own observations; the reader needs enough background information to understand the case study and follow the results.

Input The new process is introduced to the participants. The group of participants should be large enough to draw meaningful conclusions.

Output A case study produces experience reports for a process model. Best practices with a process can be detected. The reports enhance understanding rather they offer explanations.

Advantages The main advantages of the case study model is its emphasize on practitioner's experiences. It exemplifies for future user which potential experiences in applying a procedure they will make. Since case studies incorporate many different views and interests the diversity of the process can be understood. The amount and richness of available information cannot be obtained with other evaluation approaches. The view on the process is thus very broad.

Disadvantages The value of the case study description depends on the capabilities of the evaluator. As the evaluator is confronted with many influencing variables it is difficult to extract the meaningful ones. The evaluator or participant must recognize the important issues. This depends on the evaluator asking the right questions, or the participant making the right observations. In case of contradicting interests the evaluator must balance the different viewpoints. This can be best resolved in just portraying the experiences in the case study without judging them.

It is difficult to compare different case studies as they depend on the organizational setting, the involved participants and the evaluators experience.

7.1.4 Other Approaches

Additional to the evaluation approaches elaborated on in the previous section there are many others, *e.g.* the "Behavioral objectives", the "Decision making", the "Art criticism" and the "Quasi-legal" model. Our aim in the SEKT project is to provide guidelines for

Feature			METHON-TOLOGY	On-To-Knowledge (OTK)	HCOME	DILIGENT
Ontology management activities	Scheduling		Proposed	Described	NP	from OTK
	Control		Proposed	Described	NP	from OTK
	Quality assurance		NP	Described	NP	from OTK
Ontology development oriented activities	Pre development processes	Environment study	NP	Proposed	NP	from OTK
		Feasibility study	NP	Described	NP	from OTK
	Development processes	Specification	Descr. in detail	Descr. in detail	Proposed	Described
		Conceptualization	Descr. in detail	Proposed	Proposed	Descr. in detail
		Formalization	Described	Described	Proposed	Descr. in detail
		Implementation	Descr. in detail	Described	Proposed	Described
	Post development processes	Maintenance	Proposed	Proposed	Described	Descr. in detail
		Use	NP	Proposed	Described	Described
		Evolution	NP	NP	Proposed	Descr. in detail
Ontology support activities	Knowledge acquisition		Descr. in detail	Described	NP	Proposed
	■ Distributed know. acquisition		NP	NP	Proposed	Described
	■ Onto. Learning integration		NP	NP	NP	Described
	■ Partial autonomy		NP	NP	NP	Described
	Evaluation		Descr. in detail	Proposed	NP	Proposed
	Integration		Proposed	Proposed	NP	Proposed
	Configuration management		Described	Described	NP	from OTK
	Documentation		Descr. in detail	Proposed	Described	Proposed
	■ Results		Descr. in detail	Proposed	Described	Proposed
	■ Decision process		NP	NP	Proposed	Descr. in detail
	Merging and Alignment		NP	NP	Proposed	Not detailed

Table 7.1: Summary of ontology engineering methodologies adapted from [GPFLC03]

future users of the three core technologies. By means of the "Goal free" evaluation these future practitioners can select the features of a methodology most important to them and thus select the methodology suiting him best. The "case study" evaluations provide first insights what he will encounter applying the methodology, while the "Professional review" ensures that the methodology also holds to professional standards.

7.2 Goal Free Evaluation of DILIGENT

In table 7.1 we compare DILIGENT to other well known methodologies. We have adapted the categorization of [GPFLC03] separating *Ontology management activities*, *Ontology development oriented activities* and *Ontology support activities*. To the original classification we have added the aspects of **Evolution**, different **Knowledge acquisition** modes and stages during **Documentation**.

The comparison reveals that DILIGENT is well suited for ontology engineering tasks where distributiveness and change/evolution are of major concern. Further it is the first methodology which formalizes the argumentation taking place in an ontology engineering

discussion. Hence, DILIGENT should be used in cases were tracing the engineering decisions is important. This allows future users to understand different reasons which lead to the conceptualization. We think that this aspects are very important in the context of the semantic web.

DILIGENT is less adequate for use cases were consistency of the ontology is vital. Further methodological support for merging and alignment of ontologies is still not elaborated although they are support activities. DILIGENT does not itself support Ontology management and Pre development activities, since these are already well supported by other mature methodologies.

7.3 Professional review

In order to arrive at the methodology as it was presented in section 4.4 we have proceeded in several development steps. Main drivers of the development were the results of the case studies and the feedback from external reviews. In following we present the development of the DILIGENT methodology as influenced by external reviews, while we distinguish between the evolvement of the process and the development of the argumentation framework.

7.3.1 DILIGENT process evaluation

The DILIGENT process model was evaluated at varying development stages by eight experts as part of the conference reviewing process and three ontology engineering experts in affiliated institutes. The evaluation criteria were originality, impact and technical quality. Originality judges the novelty and new aspects of the proposed model. Impact refers to the influence the model will have on the community. Technical quality refers to the methods applied to validate the hypotheses.

The first version of the DILIGENT process comprised a description of the five main stages and presented initiale tool support but neither provided a detailed description of the stages nor was argumentation supported. At this point the IBIT case study lasted for 6 weeks. The reviewers appreciated the process model and the application scenario while they pointed us in the directions which we than further elaborated. They asked for an elaboration on the arbitration process to reach consensus when conflicting changes were submitted to board. The reviewers criticized the generality of the process description and the brevity of the case study. They question the ability of non-expert users to change the ontology.

The second version of the DILIGENT process comprised a detailed description of the process model on the task level and the argumentation framework. The IBIT case study lasted for 3 month. The reviewers underlined the novelty of the process model and its value for ontology development in the semantic web context in particular, since

it addresses decentralized ontology development and ontology evolution. The reviewers recommended the development of sophisticated tools and the specification of decision metrics for each process stage. As the description of the tasks did not show the parallelism of their execution order, the reviewers had the impression that the process model was very strikt and only applicable to particular application settings. Moreover, they suggested that different parts of the ontology depending on their importance for the community might have varying life cycles. The scalability of the process model was of further concern, since it is not clear how much it costs to update the local ontology and to incorporate user changes to the shared ontology if the number of users grows.

We responded to the reviewers concerns in a third and final version of the methodology. In the last version of the methodology we abstract from the defined task and define general activities, while the tasks further refine those activities for a particular application scenario. We introduce activity diagrams clarifying the execution order of the different activities. We added controlling activities and specific metrics to facilitate the decision procedure.

Open issues The supervised application of the DILIGENT methodology in a larger case study is still an open issue. The presentation of the biology case study, however, illustrates that people already build taxonomies according to a very similar process, though they did not generalized it. With our prototypes we could support the user, though there is room for improvement in future versions in particular to support the development and evolution of large ontologies with many users.

7.3.2 Argumentation framework evaluation

The DILIGENT argumentation framework was evaluated by six experts as part of the conference reviewing process an expert on argumentation structures and two ontology engineering experts in affiliated institutes. The evaluation criteria were the same as above.

We started the development of the argumentation framework with the hypothesis that a restricted set of argument types could facilitate ontology engineering discussions in DILIGENT processes. In the initial framework we defined the argumentation process and had selected the efficient argument types. The reviewers highly estimated the approach and recognized the importance for ontology engineering. They pointed us to the IBIS argumentation model established in requirements engineering and suggested the application of the argumentation framework for ontology merging.

We subsequently extended the argumentation framework with the argumentation ontology taking into account the IBIS argumentation model. The reviewers emphasized the validity argument framework's extension. They highlighted the significance of tool support.

Open issues Although we have not developed specialized tool support for the argumentation framework, it has already proven its applicability as described in the case study section. The integration of the framework in an Ontology Engineering Environment is ongoing work.

7.4 Case Study Evaluation: First Application of DILIGENT

We applied the DILIGENT process in a peer-to-peer (P2P) case study of the SWAP project[3]. In the case study up to 7 organization with up to 28 peers took part. The case study lasted for two weeks (cf. [TPSS04]). Ontologies were used to represent the local (folder) structures of each peer, whereby each peer represented a single user. On top of these local views a shared ontology was created and evolved according to the DILIGENT methodology to facilitate searching and querying over the P2P network.

Building. In the case study two knowledge engineers were involved in building the first version of the shared ontology with the help of two ontology engineers. In this case, the knowledge engineers were at the same time also knowledge providers. In addition they received additional training such that when the P2P network was up and running on a bigger scale, they were able to act as ontology engineers on the board – which they already are doing in later stages of this case study not reported here.

The ontology engineering process started by identifying the main concepts of the ontology through the analysis of competency questions and their answers. The most frequent queries and answers exchanged by peers were analysed. The identified concepts were divided into three main modules: "Sustainable Development Indicators", "New Technologies" and "Quality&Hospitality Management". From the competency questions we quickly derived a first ontology with 22 concepts and 7 relations for the "Sustainable Development Indicator" ontology. This was the domain of the then participating organizations. Recently the other modules have been further elaborated.

Based on previous experience of IBIT with the participants we could expect that users would mainly specialize the modules of the shared ontology corresponding to their domain of expertise and work. Thus, it was decided by the ontology engineers and knowledge providers involved in building the initial version that the shared ontology should only evolve by addition of new concepts, and not from other more sophisticated operations, such as restructuring or deletion of concepts.

Local Adaptation. The developed core ontology for "Sustainable Development Indicator" was distributed among the users and they were asked to extend it with their local structures. With assistance of the developers they extracted on average 14 folders. The users mainly created sub concepts of concepts in the core ontology from the folder

[3] see `http://swap.semanticweb.org/`

names. In other cases they created their own concept hierarchy from their folder structure and aligned it with the core ontology. They did not create new relations. Instance assignment took place, but was not significant. We omitted the use of the automatic functions to get a better grasp of the actions the users did manually.

Analysing. The members of the board gathered the evolving structures and analysed them. The following observations were made:

- **Concepts matched:** A third of the extracted folder names was directly aligned with the core ontology. A further tenth of them was used to extend existing concepts.
- **Folder names indicate relations:** In the core ontology a relation inYear between the concept Indicator and Temporal was defined. This kind of relation is often encoded in one folder name. *e.g.* the folder name "SustInd2002" matches the concepts Sustainable Indicator and Year[4]. It also points to a modelling problem, since Sustainable Indicator is a concept while "2002" is an instance of concept Year.
- **Missing top level concepts:** The concept project was introduced by more than half of the participants, but was not part of the initial shared ontology.
- **Refinement of concepts:** The top level concept Indicator was extended by more than half of the participants, while other concepts were not extended.
- **Concepts were not used:** Some of the originally defined concepts were never used. We identified concepts as used, when the users created instances, or aligned documents with them. A further indicator of usage was the creation of sub concepts.
- **Folder names represent instances:** The users who defined the concept project used some of their folder names to create instances of that concept *e.g.* "Sustainable indicators project".
- **Different labels:** The originally introduced concept Natural spaces was often aligned with a newly created concept Natural environments and never used itself.
- **Ontology did not fit:** One user did create his own hierarchy and could use only one of the predefined concepts. Indeed his working area was forgotten in the first ontology building workshop.

From the discussions with the domain experts we have the impression that the local extensions are a good indicator for the evolution direction of the core ontology. However, since the users made use of the possibility to extend the core ontology with their folder names, as we expected, the resulting local ontologies represent the subjects of the organized documents. Therefore, a knowledge engineer is still needed to extend the core

[4] Year is sub class of class Temporal

ontology, but the basis of his work is being improved significantly. From our point of view there is only a limited potential to automate this process.

Revision. The board extended the core ontology where it was necessary and performed some renaming. More specifically the board introduced one top level concept (Project) and four sub concepts of the top level concept Indicator and one for the concept Document. The users were further pointed to the possibility to create instances of the introduced concepts. *E.g.* some folder names specified project names, thus could be enriched by such an annotation.

Local update. The extensions to the core ontology were distributed to the users. The general feedback of the users was generally positive. However, a prolonged evaluation of the user behaviour and second cycle in the ontology engineering process is still being performed.

LESSONS LEARNED: The case study helped us to generally better comprehend the use of ontologies in a peer-to-peer environment. First of all our users did understand the ontology mainly as a classification hierarchy for their documents. Hence, they did not create instances of the defined concepts. However, our expectation that folder structures can serve as a good input for an ontology engineer to build an ontology was met.

Currently we doubt that our manual approach to analyzing local structures will scale to cases with many more users. Therefore, we are currently evaluating approaches to automatically recognizing similarities in user behaviour. Furthermore, the local update will be a problem when changes happen more often. We have so far only addressed the ontology creation task itself – we are currently measuring if users get better and faster responses with the help of DILIGENT-engineered ontologies. All this is current work.

In spite of the technical challenges, user feedback was very positive since the tool was integrated into their daily work environment and could be easily used and the tool provided very beneficial support to perform their tasks. However, it will require the introduction of some new features in order to ease ontology editing tasks by users without a knowledge engineering background.

Chapter 8

Conclusions

Given the increasing importance of knowledge in contemporary business processes, the introduction of Knowledge Management in enterprizes requires an holistic approach to it, which considers organizational, technical and human aspects in equal measure. The SEKT project focuses on technical aspects of KM, in particular on the combination of ontologies, human language technology and machine learning to automate knowledge management processes.

We develop in this deliverable the ontology engineering methodology DILIGENTOntologies build the conceptual backbone of the considered KM applications, therefore DILIGENT concentrates on the evolution, extraction and refinement of ontologies. DILIGENT comprises the steps **Build**, **Local Adaptation**, **Analysis**, **Revision** and **Local Update** and introduces a board to supervise changes to a shared core ontology.

DILIGENT incorporates detailed guidelines for all the stages defined in the methodology. The guidelines are defined on a very generic level as well as on a level tailored for a specific use case. Besides the detailed definition of activities and actions occurring in the process, DILIGENT is aligned with a parametric cost estimation model, which helps to calculate the total cost of ownership of the developed ontologies.

The DILIGENT methodology includes a fine-grained argumentation framework for developing consensual ontologies in distributed settings. The argumentation framework comprises an argumentation process and guidelines for argument provision. We describe the arguments most relevant in ontology engineering discussions and define a formal argumentation model which can be applied to trace, check and document the development of an ontology.

A process model, offering fine-grained guidance for ontology learning process, is currently under development. We present our initial ideas and expect the results of the case studies in order to detail our descriptions.

The non technical aspects are taken into consideration as well; in that we describe the HIKNOW methodology, which offers a holistic view on knowledge management, and

shows the interdependencies between different aspects of KM. As SEKT technologies provide solutions for challenging knowledge management problems, we first have to determine the current state of affairs in an enterprise in order to introduce the technologies appropriately. The HIKNOW maturity level analysis provides the means to determine the maturity of a company w.r.t. knowledge management. For the SEKT case studies we will first determine their knowledge management maturity level, and examine how they could apply SEKT technologies from their specific starting point.

The DILIGENT methodology is already applied in the SEKT case studies and initial best practices are summarized in the companion deliverable 7.2.1. The methodology will thus be extended by capturing lessons learned and best practices. In the end, the methodology will be an illustrated guidebook for implementing and applying the SEKT technology in different settings to facilitate the take-up and transfer of the technology.

Bibliography

[ACC+98] Giuliano Antoniol, F. Calzolari, L. Cristoforetti, Roberto Fiutem, and Gianluigi Caldiera. Adapting function points to object-oriented information systems. In *CAiSE '98: Proceedings of the 10th International Conference on Advanced Information Systems Engineering*, pages 59–76, London, UK, 1998. Springer-Verlag.

[ACFLGP01] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. 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*, 2001.

[Alb93] F. Albrecht. *Strategisches Wissensmanagement der Unternehmensressource Wissen*. Verlag Peter Lang, Frankfurt am Main, 1993.

[ASvE04] F. Aschoff, F. Schmalhofer, and L. van Elst. Knowledge mediation: A procedure for the cooperative construction of domain ontologies. In A. Abecker, L. van Elst, and V. Dignum, editors, *Proceedings of Workshop on Agent-Mediated Knowledge Management at the 16th European Conference on Artificial Intelligence (ECAI'2004)*, pages 20–28, Valencia, Spain, August, 22-27 2004.

[AvE04] A. Abecker and L. van Elst. Ontologies for knowledge management. In Staab and Studer [SS04], chapter 22, pages 435–454.

[B. 97] B. W. Boehm and C. Abts and B. Clark and S. Devnani-Chulani. COCOMO II Model Definition Manual, 1997.

[BEH+02] E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, and V. Zacharias. KAON – towards a large scale semantic web. In K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, editors, *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, 2002. Springer.

[Ber02] A.T. Berztiss. Capability maturity for knowledge management. In *13th International Workshop on Data-base and Expert Systems Applications (DEXA'02)*, Aix-en-Provence, France, 2002.

[BG04] J. Blythe and Y. Gil. Incremental formalization of document annotations through ontology-based paraphrasing. In *Proceedings of the 13th international conference on World Wide Web*, pages 455–461. ACM Press, 2004.

[BGDM03] S. Buckingham Shum, V. U. Gangmin Li, J. Domingue, and E. Motta. Visualizing internetworked argumentation. In Kirschner et al. [KSE03], pages 185–204.

[BHGS01] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic web. In *KI-2001: Advances in Artificial Intelligence*, LNAI 2174, pages 396–408. Springer, 2001.

[BLC96] A. Bernaras, I. Laresgoiti, and J. Corera. Building and reusing ontologies for electrical network applications. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'96)*, 1996.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001(5), 2001. available at http://www.sciam.com/2001/0501issue/0501berners-lee.html.

[BMM+94] P. C. Benjamin, C. Menzel, R. J. Mayer, F. Fillion, M. T. Futrell, P.S. DeWitte, and M. Lingineni. Ontology capture method (idef5). Technical report, Knowledge Based Systems, Inc., College Station, TX, 1994.

[Boe81] B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.

[CB88] J. Conklin and M. L. Begeman. gibis: a hypertext tool for exploratory policy discussion. In *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, pages 140–152. ACM Press, 1988.

[CC05] Matteo Cristani and Roberta Cuel. A survey on ontology creation methodologies. *Int. J. Semantic Web Inf. Syst.*, 1(2):49–69, 2005.

[CKC+99] P. Chapman, R. Kerber, J. Clinton, T. Khabaza, T. Reinartz, and R. Wirth. The CRISP-DM process model. Discussion Paper, March 1999. http://www.crisp-dm.org.

[CSSS01] J. Conklin, A. Selvin, S. Buckingham Shum, and M. Sierhuis. Facilitated hypertext for collective sensemaking: 15 years on from gibis. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 123–124. ACM Press, 2001.

[DC99] Sunita Devnani-Chulani. *BAYESIAN ANALYSIS OF SOFTWARE COST AND QUALITY MODELS*. PhD thesis, FACULTY OF THE GRADUATE SCHOOL UNIVERSITY OF SOUTHERN CALIFORNIA, 1999. `http://sunset.usc.edu/publications/dissertations/SChulani.pdf`.

[Den02] M. Denny. Ontology editor survey results (table), 2002. available at http://xml.com/2002/11/06/Ontology_Editor_Survey.html.

[Den04] M. Denny. Updated ontology editor survey results (table), 2004. available at http://www.xml.com/pub/a/2004/07/14/onto.html.

[DFv02] J. Davies, D. Fensel, and F. van Harmelen, editors. *On-To-Knowledge: Semantic Web enabled Knowledge Management*. J. Wiley and Sons, 2002.

[DJM02] J. Demey, M. Jarrar, and R. Meersman. A conceptual markup language that supports interoperability between business rules modeling systems. In Meersman et al. [MT$^+$02], pages 19–35.

[dMA03] A. de Moor and M. Aakhus. Argumentation support: From technologies to tools. In *Proc. of the 8th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2003)*, Tilburg, The Netherlands, June 1-2 2003.

[dMSF05] Adriana Pereira de Medeiros, Daniel Schwabe, and Bruno Feijó. Design rationale for model-based designs in software engineering. Monografias em Ciência da Computação 02/05, PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, 2005.

[Dom98] J. Domingue. 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*, 1998.

[DP98] T. H. Davenport and L. Prusak. *Working Knowledge – How organisations manage what they know*. Havard Business School Press, Boston, Massachusetts, 1998.

[Dru93] P. A. Drucker. *A Post Capitalist Society*. HarperCollins, New York, 1993.

[DSW$^+$00] A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa, and V. R. Benjamins. Wondertools? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 6(52):1111–1133, 2000.

[Eas91] S. Easterbrook. Handling conflict between domain descriptions with computer-supported negotiation. *Knowlege Acquistion*, 3(3):255–289, 1991.

[EM97] L. Edvinson and M. Malone. *Intellectual capital. Realizing your company's true value by finding its hidden brainpower*. Harper, New York, 1997.

[Epp01] M. J. Eppler. The Concept of Information Quality: An Interdisciplinary Evaluation of Recent Information Quality Frameworks. *Studies in Communication Sciences*, 1:167–182, 2001.

[ES04] Marc Ehrig and Steffen Staab. QOM - quick ontology mapping. In Frank van Harmelen, Sheila McIlraith, and Dimitris Plexousakis, editors, *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, LNCS, pages 683–696, Hiroshima, Japan, 2004. Springer.

[Euz95] J. Euzenat. Building consensual knowledge bases: Context and architecture. In *Proceedings of the 2nd International Conference on Building and Sharing Very Large-Scale Knowledge Bases (KBKS)*, pages 143–155, Enschede the Netherlands, 1995.

[Euz97] J. Euzenat. A protocol for building consensual and consistent repositories. Rapport de recherche 3260, INRIA Rhône-Alpes, Grenoble (FR), 1997.

[eV05] Bodo emann and Gottfried Vossen. Ontology engineering from a database perspective. In *ASIAN 2005*. Springer, 2005.

[Fel04] Alexander Felfernig. Effort estimation for knowledge-based configuration systems. In Frank Maurer and Günther Ruhe, editors, *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004), Banff, Alberta, Canada, June 20-24, 2004*, pages 148–154, 2004.

[Fen01] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, Berlin, 2001.

[FFR96] A. Farquhar, R. Fickas, and J. Rice. 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, November 1996.

[FL99] M. Fernández-López. 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, 1999.

[FLGPE+02] M. Fernandéz-López, A. Gómez-Pérez, J. Euzenat, A. Gangemi, Y. Kalfoglou, D. M. Pisanelli, M. Schorlemmer, G. Steve, L. Stojanovic,

G. Stumme, and Y. Sure. A survey on methodologies for developing, maintaining, integrating, evaluating and reengineering ontologies. OntoWeb deliverable 1.4, Universidad Politecnia de Madrid, 2002.

[FLGPSS99] M. Fernández-López, A. Gómez-Pérez, J. P. Sierra, and A. P. Sierra. Building a chemical ontology using Methontology and the Ontology Design Environment. *Intelligent Systems*, 14(1), January/February 1999.

[GF94] O. Gotel and A. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of International Conference on Requirements Engineering 1994*, pages 94–101. IEEE CS Press, 1994.

[GF95a] M. Grueninger and M. Fox. Methodology for the design and evaluation of ontologies, 1995.

[GF95b] M. Grüninger and M.S. Fox. TOVE: Manual of the Toronto Virtual Enterprise. Technical report, Department of Industrial Engineering, University of Toronto, 1995. available at http://www.ie.utoronto.ca/EIL/tove/ontoTOC.html.

[GF97] O. Gotel and A. Finkelstein. Extended requirements traceability: Results of an industrial case study. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, page 169. IEEE Computer Society, 1997.

[GK97] T. F. Gordon and N. Karacapilidis. The zeno argumentation framework. In *Proceedings of the sixth international conference on Artificial intelligence and law*, pages 10–18. ACM Press, 1997.

[GP96] A. Gómez-Pérez. A framework to verify knowledge sharing technology. *Expert Systems with Application*, 11(4):519–529, 1996.

[GP04] A. Gómez-Pérez. Ontology evaluation. In Staab and Studer [SS04], chapter 13, pages 251–274.

[GPAFL$^+$02] A. Gómez-Pérez, J. Angele, M. Fernandéz-López, V. Christophides, A. Stutt, Y. Sure, et al. A survey on ontology tools. OntoWeb deliverable 1.3, Universidad Politecnia de Madrid, 2002.

[GPFLC$^+$02] A. Gómez-Pérez, M. Fernandéz-López, O. Corcho, T. T. Ahn, N. Aussenac-Gilles, S. Bernardos, V. Christophides, O. Corby, P. Crowther, Y. Ding, R. Engels, M. Esteban, F. Gandon, Y. Kalfoglou, G. Karvounarakis, M. Lama, A. López, A. Lozano, A. Magkanaraki, D. Manzano, E. Motta, N. Noy, D. Plexousakis, J. A. Ramos, and Y. Sure. Technical roadmap. OntoWeb deliverable 1.1.2, Universidad Politecnia de Madrid, 2002.

[GPFLC03] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowlege Processing. Springer, 2003.

[GPS98] A. Gangemi, D.M. Pisanelli, and G. Steve. Ontology integration: Experiences with medical terminologies. In Nicola Guarino, editor, *Formal Ontology in Information Systems*, pages 163–178, Amsterdam, 1998. IOS Press.

[GR02] Y. Gil and V. Ratnakar. Trellis: An interactive tool for capturing information analysis and decision making. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 37–42. Springer-Verlag, 2002.

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gru95] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928, 1995.

[GW02] N. Guarino and C. Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, February 2002.

[Hal01] T. Halpin. *Information Modelling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan-Kaufmann, 2001.

[Han05] Siegfried Handschuh. *Creating Ontology-based Metadata by Annotation for the Semantic Web*. PhD thesis, Institut AIFB, Universtiy of Karlsruhe (TH), 2005. Prof. Dr. Rudi Studer and Prof. Dr. Christof Weinhardt.

[HH02] I. Horrocks and J. A. Hendler, editors. *Proceedings of the First International Semantic Web Conference: The Semantic Web (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science (LNCS)*, Sardinia, Italy, 2002. Springer.

[HJ02] C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Communications of the ACM*, 45(2):42–47, 2002.

[HK05] Mark Hefke and Frank Kleiner. An ontology-based software infrastructure for retaining theoretical knowledge management maturity models. In *1st workshop Formal Ontologies Meet Industry (FOMI 2005)*, 2005. `http://www.fzi.de/ipe/publikationen.php?id=1418`.

[HLW99] K. T. Huang, Y. W. Lee, and R. Y. Wang. *uality Information and Knowledge*. Prentice Hall, 1999.

[HNM02] C. J. Hou, N. F. Noy, and M. Musen. A template-based approach toward acquisition of logical sentences. In Musen et al. [MNS02], pages 77–89.

[Hol03a] C. W. Holsapple, editor. *Handbook on Knowledge Management 1 – Knowledge Matters*. International Handbooks on Information Systems. Springer, Berlin, Heidelberg, New York, 2003.

[Hol03b] C. W. Holsapple, editor. *Handbook on Knowledge Management 2 – Knowledge Directions*. International Handbooks on Information Systems. Springer, Berlin, Heidelberg, New York, 2003.

[Hor98] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proceedings of the International Conference on Knowledge Representation (KR 1998)*, pages 636–649. Morgan Kaufmann, 1998.

[Hou80] E.R. House. *Evaluating with validity*. Sage Publications, Beverly Hills, 1980.

[HSC02] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream – semi-automatic creation of metadata. *Expert Update, Special Issue - Intelligent Services for The Knowledge Lifecycle*, pages 20–31, 2002.

[Hun04] A. Hunter. Towards higher impact argumentation. In D. L. McGuinness and G. Ferguson, editors, *AAAI2004*, pages 275–280. AAAI Press / The MIT Press, 2004.

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the Fourth International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 353–367. Springer, NOV 2005.

[IEE84] IEEE. Ieee guide to software requirements specifications. Technical report, IEEE, 1984. ANSI/IEEE Standard 830-1984.

[IEE90] IEEE. IEEE standard glossary of software engineering terminology, 1990. IEEE Standard 610.12-1990, ISBN 1-55937-067-X.

[IEE96] IEEE. IEEE guide for developing of system requirements specifications, 1996. IEEE Standard 1233-1996.

[Jac96] R. Jacques. *Manufacturing the employee – Management Knowledge from the 19th to 21st Centuries*. SAGE Publications, London, Thousand Oaks, New Delhi, 1996.

[JBCV98] Dean Jones, Trevor Bench-Capon, and Pepjin Visser. Methodologies for ontology development. In *Proceedings of the IT&KNOWS Conference of the 15th IFIP World Computer Congress*. Chapman-Hall, 1998.

[JM02] M. Jarrar and R. Meersman. Formal ontology engineering in the DOGMA approach. In Meersman et al. [MT+02], pages 1238–1254.

[Kay03] A. S. Kay. The curious success of knowledge management. In Holsapple [Hol03b], pages 679–687.

[Kem87] C. F. Kemerer. An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, 30(5), 1987.

[KLS95] J. Krogstie, O. I. Lindland, and G. Sindre. Defining Quality Aspects for Conceptual Models. In *Proceedings of the IFIP8.1 working conference on Information Systems Concepts ISCO03: Towards a Consolidation of Views*, 1995.

[Koc00] V.P. Kochikar. The knowledge management maturity model: A staged framework for leveraging knowledge. In *KMWorld 2000*, Santa Clara, CA, 2000.

[Kor05] Maksym Korotkiy. On the effect of ontologies on web application development effort. In *Proceedings of the Knowledge Engineering and Software Engineering workshop*, Koblenz, Germany, 2005.

[KR70] W. Kunz and H. W. J. Rittel. Issues as elements of information systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.

[KSE03] Paul A. Kirschner, Simon J. Buckingham Shum, and Chad S. Carr (Eds.), editors. *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making*. Springer, London, 2003.

[KV03] K. Kotis and G. Vouros. Human centered ontology management with HCONE. In *ODS'03: Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems*, volume 71. CEUR-WS.org, 2003.

[KVA04] K. Kotis, G. A. Vouros, and Jerónimo Padilla Alonso. HCOME: tool-supported methodology for collaboratively devising living ontologies. In *SWDB'04: Second International Workshop on Semantic Web and Databases 29-30 August 2004 Co-located with VLDB*. Springer-Verlag, 2004.

[LAB+02] A. Léger, H. Akkermans, M. Brown, J.-M. Bouladoux, R. Dieng, Y. Ding, A. Gómez-Pérez, S. Handschuh, A. Hegarty, A. Persidis, R. Studer, Y. Sure, V. Tamma, and B. Trousse. Successful scenarios for ontology-based applications. OntoWeb deliverable 2.1, France Télécom R&D, 2002.

[LBB+02] A. Léger, Y. Bouillon, M. Bryan, R. Dieng, Y. Ding, M. Fernandéz-López, A. Gómez-Pérez, P. Ecoublet, A. Persidis, and Y. Sure. Best practices and guidelines. OntoWeb deliverable 2.2, France Télécom R&D, 2002.

[LE04] M. Langen and K. Ehms. Kmmm - knowledge management maturity model. Technical report, Siemens AG, 2004. `http://www.kmmm.org`.

[LET04] Steffen Lamparter, Marc Ehrig, and Christoph Tempich. Knowledge extraction from classification schemata. In Robert Meersman and Zahir Tari, editors, *ODBASE*, Lecture Notes in Computer Science, Larnaca, Cyprus, 25 - 29 October 2004. Springer.

[LG90] D. B. Lenat and R. V. Guha. *Building large knowledge-based systems. Representation and inference in the CYC project*. Addison-Wesley, Reading, Massachusetts, 1990.

[LTGP04] A. Lozano-Tello and A. Gomez-Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15(2), 2004.

[LUM+02] G. Li, V. Uren, E. Motta, S. Buckingham Shum, and J. Domingue. Claimaker: Weaving a semantic web of research papers. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 436–441. Springer-Verlag, 2002.

[MAAY03] G. Mentzas, D. Apostolou, A. Abecker, and P. Young. *Knowledge Asset Management: Beyond the Process-centred and Product-centred Approach*. Springer, 2003.

[Mae02] A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academics, February 2002.

[Mar97] D. Marcu. The rhetorical parsing of natural language texts. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, (ACL'97/EACL'97)*, pages 96–103, Madrid, Spain, July 7-10 1997.

[Men99] Tim Menzies. Cost benefits of ontologies. *Intelligence*, 10(3):26–32, 1999.

[MFRW00] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceedings of the International Conference on Knowledge Representation (KR 2000)*, pages 483–493. Morgan Kaufmann, 2000.

[MMS03] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302, Nov 2003.

[MMV02] B. Motik, A. Maedche, and R. Volz. A conceptual modeling approach for semantics–driven enterprise applications. In Meersman et al. [MT+02], pages 1082–1099.

[MNS02] M. Musen, B. Neumann, and R. Studer, editors. *Intelligent Information Processing*. Kluwer Academic Publishers, Boston, Dordrecht, London, 2002.

[MS01] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.

[MSBS03] D. L. Moody, G. Sindre, T. Brasethvik, and A. Solvberg. Evaluating the quality of information models: empirical testing of a conceptual model quality framework. In *Proceedings of the 25th International Conference on Software Engineering ICSE03*, 2003.

[MT87] W. C. Mann and S. A. Thompson. Rhetorical structure theory: A theory of text organization. In L. Polanyi, editor, *The Structure of Discourse*. Ablex Publishing Corporation, Norwood, N.J., 1987.

[MT+02] R. Meersman, Z. Tari, et al., editors. *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, 2002. Springer.

[NFM00] N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In R. Dieng and O. Corby, editors, *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)*, pages 17–32, Juan-les-Pins, France, 2000. Springer.

[NM01] N. Noy and D. L. McGuinness. 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, March 2001.

[NNM05] A. De Nicola, R. Navigli, and M. Missikoff. Building an eprocurement ontology with upon methodology. In *Proc. of 15th e-Challenges Conference*, Ljubljana, Slovenia, October 19-21st 2005.

[PB88] C. Potts and G. Bruns. Recording the reasons for design decisions. In *Proceedings of the 10th international conference on Software engineering*, pages 418–427. IEEE Computer Society Press, 1988.

[PBM05a] E. Paslaru Bontas and M. Mochol. A cost model for ontology engineering. Technical Report TR-B-05-03, Free University of Berlin, April 2005.

[PBM05b] E. Paslaru Bontas and M. Mochol. Towards a Cost Estimation Model for Ontology Engineering. In *Proceedings of the Berliner XML Days Conference*, 2005.

[PBMT05] E. Paslaru Bontas, M. Mochol, and R. Tolksdorf. Case Studies in Ontology Reuse. In *Proceedings of the 5th International Conference on Knowledge Management IKNOW05*, 2005.

[PM00] H. S. Pinto and J. Martins. Reusing ontologies. In *AAAI 2000 Spring Symposium on Bringing Knowledge to Business Processes*, pages 77–84, 2000.

[PM01] H. S. Pinto and J. P. Martins. A methodology for ontology integration. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP2001)*, pages 131–138, New York, 2001. ACM Press.

[PP02] O. Paulzen and P. Perc. A maturity model for quality improvement in knowledge management. In A. Wenn, M. McGrath, and F. Burstein, editors, *Enabling Organisations and Society through Information Systems, Proceedings of the 13th Australasian Conference on Information Systems (ACIS 2002)*, pages 243–253, Melbourne, 2002.

[PRR99] G. Probst, K. Romhardt, and S. Raub. *Managing Knowledge*. J. Wiley and Sons, 1999.

[PS04] R. Price and G. Shanks. A Semiotic Information Quality Framework. In *Proceedings of the International Conference on Decision Support Systems DSS04*, 2004.

[PSST04] H. S. Pinto, S. Staab, Y. Sure, and C. Tempich. OntoEdit empowering SWAP: a case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT). In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, *First European Semantic Web Symposium, ESWS 2004*, volume 3053 of *LNCS*, pages 16–30, Heraklion, Crete, Greece, May 2004. Springer.

[PSTS04] H. S. Pinto, S. Staab, C. Tempich, and Y. Sure. DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In Ramon López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 393–397, Valencia, Spain, August 2004. IOS Press.

[PWCC95] M.C. Paulk, C.V. Weber, B. Curtis, and M.B. Chrissis. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley Publishing Company, Reading, MA, carnegie mellon university, software engineering institute edition, 1995.

[Qui92] J. Quinn. *Intelligent Enterprise. A knowledge and service based paradigm for industry*. Free Press, New York, 1992.

[RA+03] U. Reimer, A. Abecker, , S. Staab, and G. Stumme, editors. *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, 2003. Gesellschaft fuer Informatik (GI).

[RD92] Balasubramaniam Ramesh and Vasant Dhar. Supporting systems development by capturing deliberations during requirements engineering. *IEEE Trans. Softw. Eng.*, 18(6):498–510, 1992.

[RVMS99] T. Russ, A. Valente, R. MacGregor, and W. Swartout. Practical Experiences in Trading Off Ontology Usability and Reusability. In *Proceedings of the Knowledge Acquisition Workshop KAW99*, 1999.

[SA02] Y. Sure and J. Angele, editors. *Proceedings of the First International Workshop on Evaluation of Ontology based Tools (EON 2002)*, volume 62 of *CEUR Workshop Proceedings*, Siguenza, Spain, 2002. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

[SAA+99] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. van de Velde, and B. Wielinga. *Knowledge Engineering and Management — The CommonKADS Methodology*. The MIT Press, Cambridge, Massachusetts; London, England, 1999.

[SAB+03] Y. Sure, H. Akkermans, J. Broekstra, J. Davies, Y. Ding, A. Duke, R. Engels, D. Fensel, I. Horrocks, V. Iosif, A. Kampman, A. Kiryakov, M. Klein, T. Lau, D. Ognyanov, U. Reimer, K. Simov, R. Studer, J. van der Meer, and F. van Harmelen. On-To-Knowledge: Semantic web–enabled knowledge management. In N. Zhong, J. Liu, and Y. Yao, editors, *Web Intelligence*, chapter 13, pages 278–301. Springer-Verlag", 2003.

[SAS03] Y. Sure, J. Angele, and S. Staab. OntoEdit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, LNCS(2800):128–152, 2003.

[Sch96a] U. Schneider. Management in der wissensbasierten unternehmung. In *Wissensmanagement* [Sch96b], pages 13–48.

[Sch96b] U. Schneider, editor. *Wissensmanagement*. Frankfurter Allgemeneine Zeitung, Frankfurt am Main, 1996.

[SEA+02a] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In Horrocks and Hendler [HH02], pages 221–235.

[SEA+02b] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In Horrocks and Hendler [HH02], pages 221–235.

[SG89] M.L.G. Shaw and B.R. Gaines. Comparing conceptual structures: Consensus, conflict, correspondence and contrast. *Knowledge Acquisition*, 1(4):341–363, 1989.

[SK93] Y. Sakamoto and E. Kuwana. Toward integrated support of synchronous and asynchronous communication in cooperative work: an empirical study of real group communication. In *Proceedings of the conference on Organizational computing systems*, pages 90–97. ACM Press, 1993.

[SMD02] S. Buckingham Shum, E. Motta, and J. Domingue. Augmenting design deliberation with compendium: The case of collaborative ontology design. In *HypACoM 2002: Facilitating Hypertext-Augmented Collaborative Modeling. ACM Hypertext'02 Workshop*, University Maryland, MD, 2002. Retrieved November 24, 2004 from `http://kmi.open.ac.uk/projects/compendium/SBS-HT02-Compendium.html`.

[SMJ02] P. Spyns, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. *SIGMOD Record – Web Edition*, 31(4), December '02 2002. Special Section on Semantic Web and Data Management; R. Meersman and A. Sheth (eds.); Available at http://www.acm.org/sigmod/record/.

[SMMS02] Ljlijana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic. User-driven ontology evolution management. In *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW*, Madrid, Spain, October 2002.

[SPKR96] B. Swartout, R. Patil, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*, Banff, Canada, November 1996.

[SRKR97] B. Swartout, P. Ramesh, K. Knight, and T. Russ. Toward distributed use of largescale ontologies. In *Symposium on Ontological Engineering of AAAI*, Stanford, CA., 1997.

[SS02] Y. Sure and R. Studer. On-To-Knowledge methodology. In Davies et al. [DFv02], chapter 3, pages 33–46.

[SS03] Y. Sure and H.-P. Schnurr, editors. *Proceedings of the 1st National "Workshop Ontologie-basiertes Wissensmanagement (WOW2003)"*, 2003. April 2003, Luzern, Switzerland; held in conjunction with [RA+03].

[SS04] S. Staab and R. Studer, editors. *Handbook on Ontologies*, volume 10 of *International Handbooks on Information Systems*. Springer, 2004.

[SSS+01] A. Selvin, S. Buckingham Shum, M. Sierhuis, J. Conklin, B. Zimmermann, C. Palus, W. Drath, D. Horth, J. Domingue, E. Motta, and G. Li. Compendium: Making meetings into knowledge events. In *Knowledge Technologies*, Austin, TX, March 4-7 2001.

[SSSS01] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems, Special Issue on Knowledge Management*, 16(1):26–34, January/Febrary 2001.

[Ste95] Stewart, R. D. and Wyskida, R. M. and Johannes, J. D. *Cost Estimator's Reference Manual*. Wiley, 1995.

[Ste97] T. A. Stewart. *Intellectual Capital – The New Wealth of Organizations*. Doubleday/Currency, a division of Bantam Doubleday Dell Publishing Group, Inc., 1997.

[Sur03] Y. Sure. *Methodology, Tools and Case Studies for Ontology based Knowledge Management*. PhD thesis, University of Karlsruhe, 2003.

[TPSS04] C. Tempich, H. S. Pinto, S. Staab, and Y. Sure. A case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT). In K. Tochtermann and H. Maurer, editors, *Proceedings of the 4th International Conference on Knowledge Management (I-KNOW'04)*, pages 225–232, Graz, Austria, June 30 – July 02 2004. Journal of Universal Computer Science (J.UCS).

[TPSS05] C. Tempich, H. S. Pinto, Y. Sure, and S. Staab. An argumentation ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT). In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, *Second European Semantic Web Conference, ESWC 2005*, LNCS, Heraklion, Crete, Greece, May 2005. Springer.

[TRJ84] S. Toulmin, R. Rieke, and A. Janik. *An introduction to reasoning*. Macmillan Publishing, 1984.

[TS98] J. Tennison and N. Shadbolt. APECKS: A tool to support living ontologies. In *Proceedings of the 11th Knowledge Acquisition Workshop (KAW'98)*, Banff, Canada, April 1998.

[TV03] C. Tempich and R. Volz. Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In York Sure, editor, *Evaluation of Ontology-based Tools (EON2003) at Second International Semantic Web Conference (ISWC 2003)*, October 2003.

[UCH+98] M. Uschold, P. Clark, M. Healy, K. Williamson, and S. Woods. An Experiment in Ontology Reuse. In *Proceedings of the 11th Knowledge Acquisition Workshop KAW98*, 1998.

[UG96] M. Uschold and M. Grueninger. Ontologies: Principles, methods and applications. *Knowledge Sharing and Review*, 11(2), June 1996.

[UHW+98] M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology Reuse and Application. In *Proceedings of the International Conference on Formal Ontology and Information Systems FOIS98*, pages 179–192, 1998.

[UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, Montreal, Canada, 1995.

[UKMZ98] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *Knowledge Engineering Review*, 13(1):31–89, 1998.

[VSTE05] Denny Vrandecic, York Sure, Christoph Tempich, and Michael Engler. Sekt methodology: Initial best practices and lessons learned in case studies. SEKT formal deliverable 7.2.1, Institute AIFB, University of Karlsruhe, DEC 2005.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[WPH02] R. Weerdmeester, C. Pocaterra, and M. Hefke. Knowledge management maturity model, vision deliverable d5.2. Technical report, FZI, 2002.

[YC03] Robert K. Yin and Donald T. Campbell. *Case Study Research: Design and Methods*, volume 5 of *Applied Social Research Methods Series*. Sage Publications Inc., Thousand Oaks, CA, 2003.