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

Ontology Engineering and Routing in Distributed Knowledge Management Applications

von

Dipl.-Wi.-Ing. Christoph Tempich

Tag der mündlichen Prüfung: 01.08.2006
Referent: Prof. Dr. Rudi Studer
1. Korreferent: Prof. Dr. Hagen Lindstädt
2. Korreferent: Prof. Dr. Steffen Staab

To my family.

Abstract

The dynamics of contemporary business conditions has caused a decentralization of the traditional enterprize organization and a shift towards transferring responsibilities from individuals organized in centralized hierarchical structures to autonomous teams. In particular innovative and value creating knowledge workers involved in research and development, marketing and management activities are affected by this change.

This thesis explores the implications this organizational trend has on knowledge management systems and proposes solutions to some of the arising challenges. It introduces a decentralized knowledge management system allowing knowledge workers to share knowledge in a peer-to-peer fashion. Within this system ontologies provide the conceptual backbone to represent and share the knowledge located at individual sites. We investigate such decentralized ontology-based knowledge management systems from an organizational and a technical perspective. In particular the work addresses the building and maintenance of ontologies and the ontology-based routing of queries in this distributed context.

DILIGENT proposes an ontology engineering methodology to support the building and evolution of ontologies in decentralized settings. A major concern in such an environment is the construction of consensually shared ontologies to enable knowledge exchange. With this respect the methodology aids ontology users in handling such a shared ontology, while preserving their autonomy by allowing for local variations. It includes an argumentation framework which structures ontology engineering discussions, ensuring the coherence and traceability of the process and facilitating later reuse by explicitly encoding assumptions underlying certain modeling decisions. The DILIGENT methodology is supported by tools and is empirically validated within three case studies.

REMINDIN' is an ontology-based routing algorithm demonstrating the value of shared ontologies for knowledge retrieval in peer-to-peer knowledge management systems. We examine available network topologies with respect to the requirements of a distributed knowledge management scenario, concluding that pure unstructured peer-to-peer networks meet these requirements. REMINDIN' is a routing algorithm for such network topologies. It exploits social metaphors in conjunction with semantic information in order to select and forward queries to remote peers. We tested the algorithm through simulations resembling real-world ontology-based peer-to-peer systems. This evaluation showed that REMINDIN' performs better in comparison to related approaches.

The thesis demonstrates that ontology-based distributed knowledge management is feasible from an organizational as well as technical point of view. The requirements of the

presented use cases are similar to the requirements of the Semantic Web indicating that the proposed solutions may also be of value in this broader context.

Acknowledgements

This thesis is the result of my work as a research assistant at the Institute AIFB at the University of Karlsruhe, Germany. In this time I had the chance to collaborate with many people who contributed to the completion of this thesis with their feedback and their research. Further I would like to acknowledge that this research would not have been possible without the commitment of the European Union through the projects SWAP and SEKT.

First of all, I would like to express my gratitude to my advisor, Prof. Dr. Rudi Studer, who has given me the freedom to conduct research in an innovative area and guided me towards the completion of my goals. I also thank Prof. Dr. Steffen Staab who mediated my first contacts with the Semantic Web and inspired major parts of this work with his valuable comments and ideas.

Moreover, I am grateful to all my colleges from the University of Karlsruhe for the excellent working atmosphere and the enjoyable teamwork. In particular I thankfully mention Marc Ehrig, Peter Haase and Dr. York Sure for the prolific collaboration within research projects and for the many unforgettable moments in various places of the world.

I am also much obliged to H. Sofia Pinto and Alexander Löser for the fruitful discussions and to Elena Simperl for the invaluable comments on the elaboration and the wording of this thesis.

Finally my thanks go to my family and friends who made this work possible through their understanding and support.

July 2006, Karlsruhe

Christoph Tempich

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AI	Artificial Intelligence
DHT	distributed hash table
DILIGENT	DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies
DKM	distributed knowledge management
DL	description logic
FAQ	frequently asked questions
FIFO	First In, First Out
IBIS	Issue Based Information Systems
IBL	Interest based locality
IT	Information Technology
KM	knowledge management
LNR	Local Node Repository
LRU	least-recently-used
OEE	ontology engineering environment
OE	ontology engineering
OJPK	Ontology of Judicial Professional Knowledge
OM	organizational memories
OPLK	Ontology of Professional Legal Knowledge
ORSD	ontology requirements specification document
OTK	On-To-Knowledge
P2P	peer-to-peer
PLK	professional legal knowledge

RDFS RDF Schema

RDF Resource Description Framework

REMINDIN' Routing Enabled by Memorizing INformation about Distributed INformation

RST rhetorical structure theory

TTL time to live

W3C World Wide Web consortium

WWW World Wide Web

Part I.

Foundations

Research is an activity
that contributes to the understanding
of a phenomenon.
— Kuhn (1996),
Lekatos (1978)

1. Introduction & Overview


Overview

In **this part** we provide the **foundations** of the presented thesis. We start with an introduction and an overview of the contributions. The following Chapter 2 reviews related areas introducing "distributed knowledge management" as the application field of our work and "ontologies" to formalize knowledge. We use "argumentation" theory to facilitate the development of shared ontologies in distributed systems. We present "peer-to-peer systems" to connect nodes in a distributed knowledge management scenario. This part ends with Chapter 3 presenting an implementation of a peer-to-peer-based distributed knowledge management system.

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

1.1. Motivation

Peter F. Drucker predicted in 1988 that the pervasive adoption of information technologies in every day business correlated with changes in demographics and economics, will induce a major reorganization of companies (Drucker, 1988). In order to cope with the dynamics of contemporary business conditions enterprises need to partially replace their traditional centralistic hierarchical organizational structures with more decentralized forms based on accountable autonomous teams. To date these predictions have become reality for organizational units responsible for innovation and value creation in research and development, marketing or management (Davenport, 2005). These departments primarily consist of so-called "knowledge workers", persons who

> *[. . .] have high degrees of expertise, education, or experience, and the primary purpose of their jobs involves the creation, distribution, or application of knowledge.*
>
> *(Davenport & Prusak, 1998)*

Consequently, their work mainly results in intellectual property and intangible assets.

As most of the tasks located within these novel organizational units are completed in collaboration individual efficiency and productivity scores are considerably dependant upon factors such as the corporate culture and the general work environment.

For the completion of their tasks, knowledge workers are required to

> *"... take decisions all the time, guided by the knowledge base they have access to, the corporate culture they have embraced, and the colleagues with whom they are constantly communicating."*
>
> *(The Economist, 2006a)*

The technological background of these processes is provided by so-called "knowledge management systems". They offer their users an inventory of facilities targeted at the creation, organization, storage, retrieval, and reuse of knowledge (Probst et al., 1998; Alavi & Leidner, 2001)—applications for groupware, workflow management, document and content management, information sharing, search and retrieval—assisting them during product development, process improvement, project management and human resource management (Tsui, 2003).

The decentralization trend reported at organizational level is however not accompanied by similar advances as regarding technological support; the architecture of current knowledge management systems, which are being used by decentralized autonomous teams, has a centralistic hierarchical control layout. Likewise in the case of organizational structures this is not adequate when it comes to the exchange of individual knowledge, ideas and experiences (Maier & Hädrich, 2004). Accounting for this fact Bonifacio et al. (2002) propose a paradigm shift towards distributed knowledge management, in which individual knowledge resides decentralized at the knowledge worker while the system organizes the knowledge exchange. Distributed knowledge management systems are less costly than their centralized counterparts, as they require less infrastructure investment and do not cause large set-up costs (Maier & Hädrich, 2004). They also reduce the barriers of entry for knowledge sharing as being part of individual workspaces, but these systems are associated with several technological challenges:

Knowledge sharing As knowledge creation is inherently subjective, it is likely that in a distributed scenario various ways of structuring knowledge will emerge. In order to share knowledge across the network it requires a uniform representation.

Knowledge location Before knowledge can be exchanged it is located within the network of knowledge workers.

Security and trust Shared knowledge can be sensitive and should not be accessible to everybody. Without central control of the published knowledge it is not clear to which extent available knowledge can be trusted.

This work focuses on distributed knowledge management systems and their integration in organizational structures. In particular we address aspects related to the creation and location of knowledge as part of these systems. A promising approach to structure organizational knowledge and enable knowledge access and exchange in distributed knowledge management scenarios is introduced by Fensel et al. (2003). The core of this proposal is the notion of "ontologies" understood as formal and shared understanding of a domain of interest (*cf.* (Uschold & Grüninger, 1996)). Ontologies communicate a well-defined meaning of the represented knowledge and therefore enable collaboration between knowledge workers. The communicated meaning forms the basis for mediating between *local knowledge structures*.

The realization of this scenario implies however new methodological and technological means to adequately support ontology engineering processes in distributed environments.

Technologically in (Fensel et al., 2003) knowledge workers are interconnected via a peer-to-peer infrastructure providing basic network services, such as message exchange and authentication. However, current solutions to information location in peer-to-peer systems are limited to the usage of keys and keywords for query processing and routing (Oram, 2001; Androutsellis-Theotokis & Spinellis, 2004). Given a formal representation of peer knowledge by means of ontologies these approaches can additionally take into account the semantic similarities between peers to improve the query results at data level.

1.2. Goals and Contributions of This Thesis

The challenge to technically and organizationally support the collaboration between autonomous and decentralized teams has been the starting point for a number of research projects (*cf. e.g.*, EDUTELLA[1], EDAMOK[2], SEWASIE[3], SWAP) and workshops (*cf. e.g.*, **AMKM** (van Elst et al., 2003; Abecker et al., 2004), **P2PKM** (Zaihrayeu & Bonifacio, 2004; Zaihrayeu & Robertson, 2005) and **P2PIR** (Nottelmann et al., 2005)). It has also been a topic of interest at knowledge management conferences (*cf. e.g.*, **CIKM** (Herzog et al., 2005), **PAKM** (Karagiannis & Reimer, 2004) and **IKNOW** (Tochtermann & Maurer, 2004)) and conferences related to the Semantic Web (*cf. e.g.*, **WWW** (W3C, 2004), **ISWC** (Gil et al., 2005) and **ESWC** (Bussler et al., 2005)).

The work presented in this thesis is the result of research performed within the EU project SWAP[4] and the EU integrated project SEKT[5]. The SWAP project focuses on the combination of Semantic Web languages for knowledge representation with peer-to-peer knowledge exchange for distributed knowledge management. The SEKT project focuses

[1] see `http://edutella.jxta.org/`
[2] see `http://edamok.itc.it/`
[3] see `http://www.sewasie.org/`
[4] EU IST-2001-34103 project SWAP, see `http://swap.semanticweb.org/`
[5] EU IST-2003-506826 integrated project SEKT, see `http://www.sekt-project.com/`

on the integration of natural language processing techniques, machine learning techniques and ontologies for knowledge management in general.

The thesis addresses the following research questions (RQs) arising in a distributed knowledge management setting with ontology-based knowledge representation.

1. Which system architecture is required to support distributed knowledge management?
    a) Which knowledge can be shared with a distributed knowledge management system?
    b) Which metadata are required to represent knowledge in a distributed knowledge management system?
    c) How can knowledge workers exchange knowledge with a distributed knowledge management system?
2. How can knowledge workers build and maintain consensual ontologies?
    a) Which process supports the creation of consensual ontologies in a decentralized, autonomous and rapidly changing environment?
    b) How can knowledge workers reach agreement for a consensual ontology?
    c) Can tools support such a process?
3. How can a knowledge worker locate knowledge in the distributed knowledge management system?
    a) Which requirements should an approach to find knowledge meet?
    b) Can ontologies improve the search quality and search efficiency?
    c) Which knowledge creation processes can be supported by ontology-based distributed knowledge management systems?

This thesis elaborates on these questions presenting an application, a methodology and an algorithm.

As part of the SWAP project we developed the SWAP architecture and the SWAP system as a reference implementation, *cf.* RQ 1 (Tempich et al., 2003). The SWAP systems offers services for knowledge creation from knowledge workers local information and its exchange with other nodes in a peer-to-peer network, *cf.* RQ 1a. In the SWAP system knowledge is represented in terms of the Semantic Web language RDF(S). The SWAP metadata model describes the local knowledge and forms the basis for knowledge location within the network, *cf.* RQ 1b (Broekstra et al., 2003). XAROP extends the SWAP system and is an application tailored for the special needs of knowledge sharing in virtual organizations (Tempich et al., 2004c); Bibster extends the SWAP system allowing for the

exchange of bibliographic metadata in a community of researchers, *cf.* RQ 1c (Haase et al., 2004a).

XAROP offers several ways to adapt an initial ontology according to the knowledge workers needs. The emerging heterogeneity in such a network increases very quickly making an alignment between different conceptualizations very difficult. We propose the DILIGENT methodology defining a process and activities associated with the adaptation of local ontologies and the reintegration of different local adaptations into a shared model, *cf.* RQ 2 (Tempich et al., 2006; Vrandecic et al., 2005). The shared ontology represents the consensual knowledge in the team, while local ontologies represent individual knowledge. The process consists of five stages which take into account the special needs of decentralized, autonomous teams working in a rapidly changing environment , *cf.* RQ 2a. In contrast to previous work DILIGENT moves the knowledge worker in the center of consideration. In order to achieve a censual ontology the knowledge workers discuss their change proposals within an argumentation framework customized to structure ontology engineering discussions and to streamline the agreement process, *cf.* RQ 2b (Pinto et al., 2004b; Tempich et al., 2005a).
We identify the requirements for tool support and show with prototypical implementations the feasibility of the approach to ontology engineering, *cf.* RQ 2c.
The methodology has been successfully evaluated in three case studies demonstrating that it supports the development of consensual ontologies for distributed and autonomous groups confronted with a rapidly changing environment (Pinto et al., 2004a; Tempich et al., 2005c).

The shared ontology provides a basis for the efficient location of knowledge in a distributed knowledge management system, *cf.* RQ 3. Guided by the requirements of distributed knowledge management we select the decentralized and unstructured approach to routing queries in the peer-to-peer network, *cf.* RQ 3a. REMINDIN' is a routing algorithm for this kind of networks (Tempich et al., 2004b). We show that REMINDIN' routes complex queries efficiently in static and dynamic peer-to-peer networks, *cf.* RQ 3b (Löser et al., 2005b; Tempich et al., 2005b). An indirect result of the routing process is the emergence of peer clusters with similar interests. People with shared interests are introduced to each other and they can share knowledge personally, *cf.* RQ 3c (Schmitz et al., 2004; Löser et al., 2005a).

The routing algorithm has been evaluated with a simulation framework using three different data sets, one of them from observations of a real world application.

1.3. Reader Guide


Overview

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

Part I provides the **foundations** for this thesis. In this chapter the contributions of this thesis are motivated and introduced. Additionally we summarize **related areas** to the presented research in order to suite the contribution into a broader context (*cf.* Chapter 2). This thesis develops a methodology and an algorithm to facilitate knowledge sharing and creation for distributed knowledge management scenarios. The underlying **application scenarios** and their **requirements** are described in Chapter 3. This chapter also introduces the **SWAP system** (SWAPSTER) enabling the connection between users willing to share knowledge on a technical level.

Part II develops the **DILIGENT methodology**. First, a **general process** is elaborated towards a well defined methodology for building a shared ontology which can be applied in real world scenarios (*cf.* Chapter 4). The methodology identifies the major roles, process stages and activities; for the process stages it identifies decision criteria to allow for a well defined transition between the single stages and their activities. Applying the methodology in decentralized environments structures the development and evolution of a shared ontology for all participants. We pay special attention to communicative interactions of the involved participants with respect to the **arguments** they exchange. Ontology engineering discussions are structured according to the argumentation sub process. The argumentation process suggests the restriction of allowed arguments in order to speed up the consensus building process and to increase the level of agreement with the shared ontology. Some activities defined in the methodology are supported by **tools**.

We elaborate on the **evaluation** of DILIGENT in Chapter 5. DILIGENT has been evaluated according to three complementary evaluation methodologies: the **goal free** evaluation, the **professional review** and in **case studies**. The goal free evaluation compares DILIGENT with established ontology engineering methodologies and results in the selection of application scenarios for which DILIGENT is particularly suited. The professional review elaborates on the evolution of the DILIGENT methodology itself and points out the strength and weaknesses of the methodology from an ontology engineering experts point of view. The three case studies describe experiences with the application of the methodology and show its usability for selected use cases.

Part III develops the **REMINDIN' query routing algorithm** (*cf.* Chapter 6). The organizational and security requirements of distributed knowledge management applications

are best met by completely decentralized and unstructured peer-to-peer networks. REMINDIN' builds on a number of **social metaphors** observed and deployed in human social networks to obtain contacts to knowledgeable persons. The social metaphors are translated into a number of **semantic overlay layers** for the peer-to-peer network which are then deployed by REMINDIN' to find information efficiently. In particular does REMINDIN' build on semantic relationships defined in the ontology to enhance its search quality.

We evaluate REMINDIN' in Chapter 7 according to established **evaluation** criteria in simulation experiments. We use three different data sets and compare REMINDIN's performance to state of the art routing algorithms and show its superior performance. We pay attention to knowledge management inspired evaluation criteria and demonstrate that REMINDIN' supports the creation of knowledge communities.

Part IV references **Related Work** on applications for **distributed knowledge management**, **ontology engineering methodologies**, **arguments** in ontology engineering discussions and **routing algorithms** (*cf.* Chapter 8). We summarize the main **contributions** and give an outlook on **future work** in Chapter 9.

Part V contains additional material in an **Appendix**, *e.g.*, a detailed description of the SWAP metadata model.

2. Review of Related Areas

Overview

In this chapter we review the areas which build the foundations for our research. We start with a definition of **knowledge management** focusing on **distributed knowledge management**. **Ontologies** are considered as a means to represent knowledge in a formal, machine readable way, thus, they can provide the conceptual backbone for a knowledge management systems. **Ontology engineering** offers guidelines to develop and maintain ontologies. Ontology engineers reach agreement in an ontology development process exchanging **arguments**. We summarize approaches to analyze general discussion processes w.r.t. exchanged arguments.
The **peer-to-peer** paradigm offers one way to implement distributed knowledge management applications; we examine existing systems with an emphasis on **semantic peer-to-peer systems**. Different **query routing** algorithms have been proposed to efficiently route queries in peer-to-peer applications.

2.1. Knowledge Management

> *The productivity of knowledge and knowledge workers will not be the only competitive factor in the world economy. It is, however, likely to become the decisive factor, at least for most industries in the developed countries.*
>
> *(Drucker, 1997)*

In recent years the importance of knowledge has been recognized as an essential factor spurring innovation and growth of companies and the economy as a whole (Leonard, 1995). While most people agree on knowledge relevance it is difficult to define it precisely. For our work we follow the definition given below, as it emphasizes the significance for "organizations" and its location.

> *Knowledge is a fluid mix of framed experience, values, contextual information, expert insight and grounded intuition that provides an environment and framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organizations,*

> *it often becomes embedded not only in documents or repositories but also in organizational routines, processes, practices and norms.*
>
> *(Davenport & Prusak, 1998)*

Knowledge is specifically different from information and data. They are located on a continuum with increasing semantics *cf.* (Probst et al., 1998, p. 36). For this thesis we assume that "knowledge is information combined with experience, context, interpretation, and reflection" (Davenport et al., 1998), while information is data with a meaning.

The knowledge written down in documents is referred to external or explicit knowledge, while knowledge contained in processes and routines, *viz.* in persons heads is referred to as implicit or tacit knowledge (*cf.* Polanyi (1966) for a philosophical distinction between the two). Nonaka & Takeuchi (1995) studied the interactions between the two forms of knowledge and introduced a spiral model describing the four possible transitions:

Socialization The process that transfers tacit knowledge from one person to another person by observation is called socialization. Tacit knowledge from the "teacher" is the source for tacit knowledge of the "learner".

Internalization Internalization refers to the knowledge transfer process from external knowledge, such as documents, to tacit knowledge held by the individual.

Combination Combination refers to the process of creating new external knowledge from external knowledge. This is the process were information technology can add the most value, because externalized knowledge can be disseminated electronically.

Externalization In the externalization process tacit knowledge is transformed into explicit knowledge.

The management of knowledge in organizations is referred to as knowledge management and is defined as the administration of cultural, organizational, human and technical aspects related to the creation, maintenance and transfer of knowledge in its various forms (*cf.* (Albrecht, 1993; Schneider, 1996)).

Probst et al. (1998) has identified the knowledge management core processes as depicted in Figure 2.1.

Knowledge Identification The goal of the knowledge identification process is to characterize and specify the knowledge relevant for the organization.

Knowledge Acquirement The relevant knowledge is gathered in the knowledge acquirement process.

Knowledge Development The objective of the knowledge development process is to enhance, combine the acquired knowledge and create new knowledge.

Knowledge Distribution The existing knowledge is made available and distributed to the members of the organization in the knowledge distribution phase.

Knowledge Use The existing knowledge is used to facilitate the core processes of the organization.

Knowledge Preservation Knowledge preservation deals with the long term storage of the created knowledge.

According to Probst et al. (1998) the six core processes interfere with each other and cannot be management independently. They depend on the strategic goals of the organization which influence the knowledge goals, and they are subject to evaluation in the knowledge audit process. Moreover, all of the core processes are influenced by cultural, organizational, human and technical considerations.

From a **cultural** perspective an organization should create an environment where knowledge sharing has a positive connotation. When organizations first introduced knowledge management the main challenge was to convince its members that they gain from knowledge sharing (Probst et al., 1998). Asking for knowledge was seen as evidence for incompetency and providing knowledge meant to give away an asset without receiving compensation. Without a culture of knowledge sharing no knowledge transfer can be initiated.

Organizational aspects of the knowledge management process relate to the definition of responsibilities and positions within the overall management structure. The organization taking knowledge management seriously provides the knowledge manager with the authority and the means to enforce the necessary organizational activities. It further ensures that knowledge management related activities are not an additional task without compensation, but an integral part of the daily work of an employee.

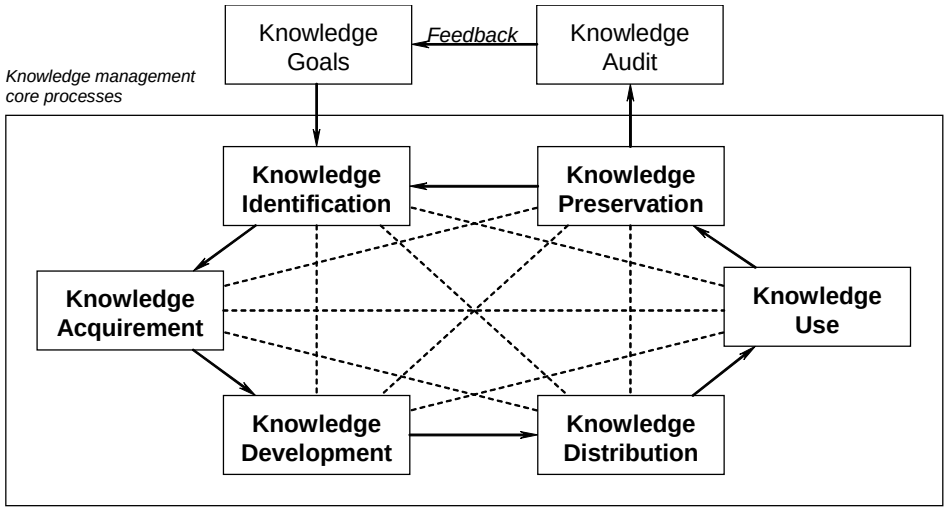


Figure 2.1.: Knowledge Management Core Processes

People should be in the center of any knowledge management effort as they are the knowledge providers. The organization should provide incentives to share knowledge and expertise should be appreciated. Furthermore, it is important to note that creation of knowledge requires imagination and creativity. Knowledge is a subjective matter and thus often related to the experience of an individual and open to interpretation. This should be considered implementing technical knowledge management support systems.

Knowledge management from a **technical** perspective – the standpoint of this thesis – is concerned with the provision of support technology for each core process. Information technology in the form of, *e.g.*, organizational memories, collaborative work environments or data mining, is used for knowledge management (Tsui, 2003). Organization memories provide access to unstructured, semi-structured and structured information available in the organization; collaborative work environments facilitate, *e.g.*, the instant communication between the members of the organization, and data mining techniques are used to unveil knowledge hidden in huge amount of data. In the following we will focus on organizational memories supporting knowledge distribution and preservation.

Although the knowledge management processes do not advocate any architecture for organizational memories, we predominantly find centralized ones. Knowledge is published and stored according to a centrally defined structure in a centrally organized repository (D. ÓLeary, 1998; Davenport et al., 1998). Knowledge providers add metadata to the published knowledge according to the predefined structure in order to facilitate knowledge retrieval. The process of knowledge provision is well defined. The knowledge can be accessed though an enterprise knowledge portal. The main advantages of enterprise knowledge portals are: (i) available knowledge can be accessed at a predefined service level, (ii) the quality of the published knowledge can be supervised in predefined publishing processes, (iii) knowledge is published according a common structure and (iv) access control mechanisms can be enforced.

Nevertheless, there are also some disadvantages with this solution for knowledge sharing (Maier & Hädrich, 2004). It incurs additional effort for publishing and maintenance on the knowledge provider side without any direct benefit. Organizations report, that the provided knowledge is not updated regularly, that not all required metadata is filled-in, and that not all knowledge is published. It is further observed that the centrally defined structure does not always suit the knowledge provider, or he may misinterpret it leading to wrong classifications w.r.t. the global structure . Another issue is cost as discussed in (Schmücker & Müller, 2003). The creation and maintenance of a centralized knowledge management system is related to a large investment effort in terms of time and money. Time is invested in the creation of a global structure and the knowledge provision processes and money in the system infrastructure. Furthermore, Bonifacio et al. (2002) argue that centralized knowledge management systems take on a traditional managerial control paradigm, which stands against the subjectivity and sociality as essential features of knowledge creation and sharing.

In particular the infrastructure investment and the setup costs make centralized knowledge management solutions unattractive for, *e.g.*, virtual organizations, small companies or projects. None of the members of a virtual organization would take the responsibility to maintain the central solution as it may leave the consortium before the overall project is finished. Small companies can often not effort a centralized solution, and projects may not have a sufficiently long duration that the initial effort can be outweighed. For these scenarios Bonifacio et al. (2002) propose an alternative solution, *viz.* distributed knowledge management.

2.1.1. Distributed Knowledge Management

Bonifacio et al. (2002) begin their analysis of the knowledge creation process at an epistemological level and compare the assumptions of centralized knowledge management solutions with the nature of knowledge. They conclude that centralized solutions require that knowledge can be created according to predefined processes, its creation is controllable and that it can be captured in an objective and predefined corporate language. Knowledge, however, is inherently subjective and its related to the context and perspective of the person creating it. This contradicts the assumptions underlying centralized solutions to knowledge management.

Distributed knowledge management (DKM) as opposed to centralized knowledge management leaves the knowledge at the places where it is created. Each person manages the knowledge autonomously and organizes it heterogeneously. Distributed knowledge management systems accept this and offer solutions to allow for coordination and interoperability between the different knowledge providers. The knowledge is thus created and managed bottom-up instead of created bottom-up and managed top-down. In DKM scenarios shared structures emerge automatically when people collaborate and discuss their own perspectives with their peers.

Schmücker & Müller (2003) report on experiences in the application of DKM systems. They identify some of the main advantages of this approach to knowledge sharing in comparison to centralized solutions. DKM systems ensure instant access to up-to-date knowledge, knowledge providers have no publishing effort, they can follow a self-determined organization of knowledge and people have direct access to other person knowledge through direct communication. The last aspect is particularly important as DKM system support indirectly the *socialization* knowledge transfer process in this way as people can communicate directly. Although possible in centralized solutions, it comes more natural in the decentralized setting.
They also emphasize that DKM solution do not require additional costs for a central infrastructure, and that they observe less duplication in form of documents stored locally and on central servers.

Nevertheless, they also point out a number of disadvantages of DKM systems. Finding of information becomes more difficult as their is no shared structure to organize it. More-

over, the same document can be found in different places, and it can be available in many version. This implies that there is no guarantee for the quality of the found information. It is inherent to peer-to-peer (P2P) systems that not all the information is always available, which may reduce the confidence in such a system. People are also worried about uncontrolled access to their local knowledge.

From this general analysis and experiences we can derive general requirements on DKM systems.

2.1.2. Distributed Knowledge Management Systems

Requirement	Description
Infrastructure level	
Access	DKM systems should provide access to local knowledge.
Security	Access should only be provided to trusted parties as knowledge may be very sensitive.
Application level	
Local perspective	Local knowledge can be organized according to the local subjective view.
Owner ship model	In centralized systems it is possible to enforce a predefined publishing process in order to guarantee quality standards and trace the ownership of published content. In DKM systems the possibility to determine the ownership of content is more difficult, but equally important to establish a certain level of trust in the found content.
Content-based search	In contrast to file sharing networks in DKM networks searching for file names is not sufficient. Search should be based on content.
Localization	Relevant knowledge should be retrievable within large P2P networks.
Community level	
Formation	DKM systems should allow for community formation.
Community perspective	The creation of a community perspective on the shared knowledge should be possible.
Interoperability	The DKM system should bridge between different local perspectives.
Free riding	In file sharing systems free riding is observed, *i.e.*, the phenomenon that people download many files, but do not share any. In the case of knowledge this is more problematic as knowledge is a major asset for the carrier of an individual and free riding may thus undermine the knowledge sharing culture.

Table 2.1.: Requirements for Distributed Knowledge Management Systems

Without loosing generality we use in this thesis a P2P infrastructure to connect the knowledge providers with each other. For P2P DKM systems Schoder & Fischbach (2003) proposes a separation in three system levels: infrastructure, application and community level. The first provides basic mechanisms for communication, security, resource identification and peer identification. The second provides services to the users supporting their

process needs. The community level comprise the social activities which are enabled by the P2P paradigm.

Requirements In order to compile requirements on DKM systems we rely on Bonifacio et al. (2002) analysis from a sociocultural perspective and on Susarla et al. (2003) analysis comparing existing P2P file sharing systems. We categorize the requirements according to the three system levels in Table 2.1.

Implementations Current P2P applications for knowledge management offer the basic functionality to access remote peers files, support collaboration and synchronization, allow for messaging and offer email organization (*cf. e.g.*, (Tsui, 2001; Wegner, 2002)). They implement a security mechanism based on a central authentication server. In the market of small and medium sized companies an owner ship model is not required, as most people still know each other. Most systems offer keyword based search for document retrieval. Bonifacio et al. (2004a) propose a DKM system called KEEx, which implements a context model allowing each peer to create its own context and to organize its knowledge accordingly (*cf.* Section 8.1). In their model localization is based on manual definition of different peer groups which can be searched. Other implementations maintain a central server with index information for all peers in the network. On the community level the KEEx system allows for the manual creation of groups interested in a specific context. They provide a mapping algorithm in order to align contexts of different peers. Free riding is not an issue in the market of current DKM systems.

In this thesis we follow the approach to knowledge representation proposed in Fensel et al. (2003). We use ontologies in order to structure and describe the knowledge available on the peers. In Section 3 we present the SWAP system providing basic services on the infrastructure level. In Part II we present an ontology engineering methodology which structures the development process for local perspectives and community perspectives. Part III pays attention to the localization of knowledge in large P2P systems.

2.2. Ontologies

The term 'Ontology' (Greek. onto = *being*, logos = *to reason*) was first use by Aristotle to describe "the science of being *qua* being". *Categories of being* explain and classify everything what exists. The idea was to deduce from a few universals all existing matter (Störig, 1992, p. 587). Philosophers as Nicolai Hartmann, however, postulate that the world is neither totally recognizable nor totally unrecognizable, but it is only "*representational* recognizable" (Störig, 1992, p. 587).

In this thesis, though, we are more interested in the 'ontology' with a lowercase 'o' (computer science) than with the upper case 'O' (philosophy) (*cf.* (Guarino, 1998a)). In computer science the following definitions for ontologies are most cited:

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

This definition emphasizes the *explicit* specification, which make ontologies interesting for machine processing. For this thesis we want to highlight also another aspect of ontologies, *i.e.*, they are *shared* specifications.

> *'Ontology' is the term used to refer to the shared understanding of some domain of interest...*
>
> *(Uschold & Grüninger, 1996)*

In the community the combined phrase cited hereafter is therefor very popular:

> *An ontology is a formal explicit specification of a shared conceptualization.*
>
> *(Uschold & Grüninger, 2004)*

In this sense an ontology is a *conceptualization* as it refers to an abstract model of how people think of things in the world. An ontology conceptualizes usually only a specific domain of interest. It is an *explicit specification*, because it provides names and definitions for the concepts and relationships given in the abstract model. A definition says how a term relates to other terms. *Formal* expresses that the ontology can be encoded in a representation language which is machine readable and has formal semantics. As mentioned above for this thesis the *shared* aspect is very important. An ontology is a conceptualization which more than one individual has agreed on. The ontology can thus be used and reused across different applications and communities (*cf.* (Uschold & Grüninger, 2004)).

Depending on the application and the purpose of an ontology one can classify an informal catalog of terms as an ontology as well as a very formal list of general constraints (*cf.* (McGuinness, 2003)). Although this spans a wide range for the application of ontologies, we can still identify some general elements an ontology consists of, *i.e.*, the general ontology entities. An ontologies usually consists of a list of terms describing the things of interest in the domain. These things are the classes (also called concepts) of the ontology and they are arranged in a taxonomy spanning a subclass-of hierarchy. Each class can be defined more specifically using PROPERTIES (also called SLOTS or ROLES) and attributes. It is also possible to put restrictions on the properties using *role restrictions* (also called *facets*). More formal ontologies contain also axioms specifying the meaning of a concept further. If an ontology is represented in a description logic (DL) classes, properties and axioms are also referred to as the TBox of an ontology. Concrete instances (also called individuals) of classes plus the TBox constitute a *knowledge base*. The instances of an DL ontology are also referred to as its ABox.

Representation languages for ontologies A number of formalisms are available to represent ontologies in a machine readable way. In the CYC project for example an ontology representation language was defined in order to model the world knowledge (Lenat & Guha, 1990). The frame based ontology representation language F-Logic was proposed by (Kifer et al., 1995).

This thesis builds on ontology representation languages which were defined in the Semantic Web context (Berners-Lee et al., 2001). 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 cooperation"(W3C, 2001). One result of the Semantic Web initiative is a stack of representation languages to formalize ontologies (Berners-Lee, 2000). OWL-DL, for example, follows the description logic paradigm to represent knowledge (Dean et al., 2002). OWL is the latest recommendation of the World Wide Web consortium (W3C) for an ontology language. In the SWAP system knowledge is represented following the semantics of RDF(S) (Lassila & Swick, 1999). Although it offers less expressivity than OWL it is sufficient to demonstrate the benefits of ontologies for DKM systems.

Differences to other modeling paradigms Ontologies are not the only mechanism to conceptualize real world objects. The database community uses, *e.g.*, entity relationship (ER) models (Chen, 1976) in order to specify the database design. Devedžić (2002) and Uschold & Grüninger (2004) extensively analyze the differences and similarities between, *e.g.*, ontology modeling and object oriented analysis and design, ontology modeling and traditional software engineering and ontology modeling and component-based software engineering. We exemplify the result of the analysis with a comparison to the ER model. Depending on the complexity of the representation language used to formalize the ontology it is possible to formalize the semantics of an ER model with little losses as an ontology and vice versa.[1] But the differences start were the pure formalism ends. An ontology is used to describe a domain as complete as possible, to give its concepts a specific meaning. The ontology is used for communication, therefore inherently a shared model and more concrete specifications reduce the possibility of misinterpretations. In comparison a database design optimizes the performance for the desired application, paying little attention to the completeness of the underlying model w.r.t. to the domain beyond the application needs.
Without providing a full comparison of the different paradigms we conclude that the difference between ontologies and other conceptualization mechanisms lies predominately in the purpose of its use. Ontologies are used for communication and meaning clarification between different parties, while other paradigms usually focus on application design and efficiency. This is also reflected in the use cases for ontologies.

[1] Note, that an ER model does not *per se* provide formal semantics, but most modeling primitives may be aligned with ontology primitives in a straightforward manner.

2.2.1. Ontology Use in Information Systems

Many different types of applications use ontologies to a various extent as a conceptual backbone. Jasper & Uschold (1999) distinguish four different types of ontology based systems:

Neutral Authoring An enterprise with many applications can use a common ontology as the semantic core for all its data structures. The idea is to translate the ontology in the target application formats and use it as a basis for further development. Although the initial effort for the ontology development can be huge, the enterprise gains advantages in the form of reusable knowledge, increased maintainability and long term knowledge retention.

Common Access to Information Views are an established way to facilitate information integration if the information is modeled according to different data schemas (*cf.* (Ullman, 2000)). An ontology can represent the global view. Many standardization efforts are underway to create uniform descriptions for, *e.g.*, product data. As ontologies allow for a detailed description of the semantics underlying different data sources, they can be used to check their consistency (*cf.* (McGuinness, 2003)). In some cases, however, simple use of an ontology as a controlled vocabulary is already helpful.

Ontology-based Specification The specification of software is another application field for ontologies. Requirements can be characterized and specified using ontologies. Following the ideas of the Model Driven Architecture promoted by the OMG[2], ontologies can assist the validation and verification of software (Uschold & Grüninger, 2004). Due to the formal specification of an ontology, changes in the model can be directly propagated to the implementing software.

Ontology-based Search An ontology can provide an abstract description for various information repositories and thus assist search. McGuinness (2003) highlight that ontologies can support site organization and navigation, browsing and allow for structured, comparative, and customized search. Search gains from the exploitation of the encoded generalization/specialization information and the sense disambiguation provided by an ontology.

In the case of, *e.g.*, product search ontologies offer configuration support and completion. In this cases complex features requests from customers are characterized in the ontology to the level of detail necessary to select from the available products.

Search is also the application scenario for ontologies in knowledge management systems (D. ÓLeary, 1998; Fensel, 2001; Abecker & van Elst, 2004). Ontologies are well suited for knowledge management applications as they provide a shared

[2] `http://www.omg.org/mda/`

> conceptualization, fitting to Davenport & Prusak (1998) postulation that *"people can't share knowledge if they don't speak a common language."*

The use cases mentioned in this thesis take advantage of ontologies supporting search and interoperability and exploit generalization/specialization information.

Although ontologies are beneficial for a number of scenarios there is one major obstacle to their fast and wide spread adoption in information systems: It is complicated and time consuming to build and maintain ontologies. The process to develop ontologies has therefore attracted attention and a number of methodologies were proposed to organize the ontology engineering process.

2.2.2. Ontology Engineering Methodologies

Ontology Engineering is formally defined as

> *"the set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies."*
>
> *(Gómez-Pérez et al., 2003)*

In this thesis we focus on the methodological aspects of ontology engineering.

2.2.2.1. Terminology

The first experiences in building ontologies have lead to the conception of different methodologies to support this process. The first methodologies described particular experiences building ontologies for specific applications (*cf.* (Grüninger & Fox, 1995; Uschold & Grüninger, 1996)).[3] However, none establishes a precise definition for the terms *methodology, method, technique, process, activity* in order to describe the engineering effort (de Hoog, 1998). We will follow the terminology introduced by the IEEE for software development methodologies as it is related to ontology engineering (*cf.* (Gómez-Pérez et al., 2003)).

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"*(IEEE, 1990).

A **method** is a set of *"orderly processes or procedures used in the engineering of a product or performing a service"*(IEEE, 1990). A **technique** is *"a technical and managerial procedure used to achieve a given objective"*(IEEE, 1990).

[3] See Section 8.2.1, page 212, for a more detailed discussion of those works.

A **process** is a *"function that must be performed in the software life cycle. A process is composed of activities"* (IEEE, 1996). An **activity** is *"a constituent task of a process"* (IEEE, 1996). A **task** *"is a well defined work assignment for one or more project members. Related tasks are usually grouped to form activities"*(IEEE, 1996).

The distinction between activity and task is not strikt: Depending, *e.g.*, on the size of the engineering setting a task which may be performed in one step for a small ontology may be an activity in other cases.

2.2.2.2. Ontology Engineering Activities and Roles

The activities defined in established ontology engineering methodologies can be grouped according to three categories: **Ontology management activities**, **ontology development oriented activities** and **ontology support activities** (Gómez-Pérez et al., 2003). Figure 2.2 visualizes the interdependencies of the main activities.

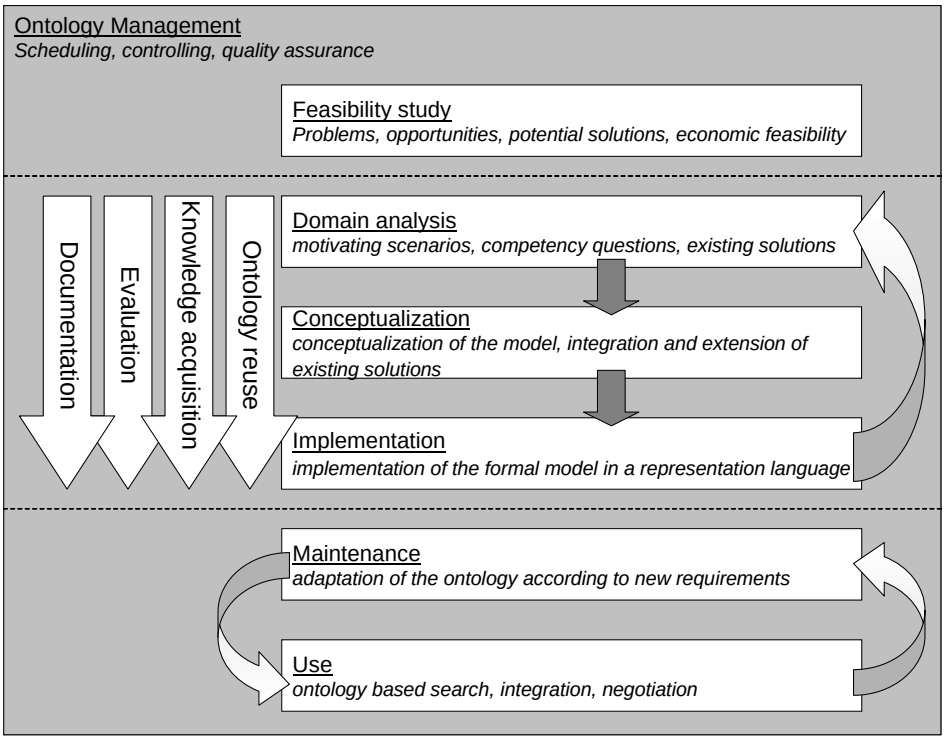


Figure 2.2.: Core Ontology Engineering Activities

Ontology management activities Ontology management contains the activities related to the organization of the ontology development process. *Scheduling*, *controlling* and *quality assurance* are management activities. The identification of required tasks with their

respective durations and their arrangement on a time schedule is part of the scheduling activity. The schedule is monitored as part of the controlling activities. The quality assurance activities guarantee that the resulting products of the development activities, such as the ontology, software and documentation achieve the desired quality level.

Ontology development oriented activities The ontology development oriented activities group the activities concerned with the development of the ontology. They are separated into pre-development, development and post-development activities. In the pre-development phase the *environmental study* and the *feasibility study* examine if an ontology based application or the use of an ontology in a given context is the right way to solve the problem at hand. *Specification*, *conceptualization*, *formalization* and *implementation* are classical ontology development activities. The *maintenance* and *use* of the ontology are post-development activities. We describe these activities in more detail in Section 4.3.1.

Ontology support activities A collection of ontology support activities assist in building the ontology and are performed in parallel to the development activities. Gómez-Pérez et al. (2003) lists *knowledge acquisition*, *evaluation*, *reuse*, *integration*, *merging*, *documentation*, *alignment* and *configuration management* as ontology support activities. We add *argument provision* to this list of support activities. The objective of the knowledge acquisition activity is the elicitation of the relevant domain knowledge from experts or other sources. The (semi-)automatic acquisition of knowledge from electronic sources is referred to as ontology learning (Maedche & Staab, 2001). In the evaluation process the results of the development activities are reviewed w.r.t. their quality. Integration and merging specify the processes to build ontologies reusing existing ones. The documentation process organizes the description of the results of the development activities. Alignment is the process of providing mappings between different ontologies, which formalize overlapping domains. The configuration management activity maintains the different versions of an ontology and its documentation. Finally, argument provision captures the arguments in favor and against design decisions.

The activities defined in an ontology engineering methodology are carried out by actors taking different roles in the process.

Roles Classical ontology engineering methodologies introduce three different roles: *Knowledge engineer*, *ontology engineer* and *domain expert*. The knowledge engineer and ontology engineer elicit the domain knowledge from the domain expert. The domain expert knows for the domain of interest the relevant concepts and the interdependencies between them; he can point to further sources of knowledge, *e.g.*, in the literature. The 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 the same person taking the role of the knowledge engineer and ontology engineer.

Additionally to these classical roles it is the role of the *user* to utilize the ontology for his needs.

The importance of the ontology engineering activities within a specific methodology depends on, *e.g.*, the use case for the ontology-based application, the complexity of the built ontology, the available information sources and the experience of the ontology engineers.

Staab et al. (2001), Staab et al. (2003) propose an ontology engineering methodology for the ontology development for centralized knowledge management systems. In this case the knowledge management meta-process (ontology engineering process) is orthogonal to the knowledge management processes. The knowledge management meta-process structures the development of an ontology for the knowledge management application, while the knowledge management process structures the knowledge creation and use processes. As detailed in Section 8.2 the process consists of the five phases "Feasibility Study", "Kickoff", "Refinement", "Evaluation" and "Application & Evolution". The "Kickoff" phase for example combines *scheduling*, *specification*, *knowledge acquisition* and *integration* activities. It involves the three classical ontology engineering roles.

In this thesis we develop an ontology engineering methodology for the creation and maintenance of ontologies in DKM applications.

2.3. Argumentation

Engineering an ontology is a social process. The participants in an ontology engineering effort have slightly different views on the world thus harmonization requires the discussion of modeling decisions. During the discussion, participants exchange arguments which may support or object to certain ontology engineering decisions. Experience from Software Engineering shows that tracking exchanged arguments can help users at a later stage to better understand the assumptions underlying the design decisions (Conklin & Begeman, 1988; Selvin et al., 2001; Ramesh & Dhar, 1992). Furthermore, as the constructed ontology becomes larger, ontology engineers might argue in a contradictory way without knowing so.

This thesis analyzes the arguments exchanged in ontology engineering discussions in order to better understand the communicative behavior of the participants building an ontology. There are a number of argumentation theories ranging from informal explanations of argumentation threads to very formal specifications, which can be applied to perform this analysis. In the following we introduce the oldest model of natural argumentation, *viz.* the Toulmin model, in order to motivate the idea. For the analysis of ontology engineering discussions we have selected the rhetorical structure theory as it provides a very detailed analysis framework. Additionally we introduce the IBIS model – a very general and established method – which will serve as a basis for our argumentation framework.

2.3.1. Toulmin Model

The Toulmin model is one of the first models, that try to explain how real people argue. It has three main components: **Data**, **Claim** and **Warrant**. **Data** refers to the facts, data and information, which are the reason for claim (Toulmin, 1958; Toulmin et al., 1984). It is important that all participants in the discussion agree on the data, as the data form the ground of any serious discussion. Data is used in order to give evidence to the claim made. **Claim** refers to the position on an issue or the conclusion being advocated. A claim is a statement one persons asks another one to accept. The **warrant** serves as the logical connection between the data and the claim, thus it represents the reasoning process used to arrive at the claim. The warrant establishes the relevance of the data for the claim. Other components are the **backing**, which provides material support for the warrant, the **reservation** relates to the exceptions to the claim and the **qualifiers** strengthens the claim in relation to other claims.

The Toulmin model is general, but does not lend itself easily to a formalization. For this reason the IBIS was defined.

2.3.2. Issue Based Information Systems

The Issue Based Information Systems (IBIS) model was introduced by Horst Rittel and colleagues during the early 1970's (Kunz & Rittel, 1970). IBIS was developed to provide a simple yet formal structure for the discussion and exploration of "wicked" problems. A problem is wicked, as opposed to "tame", if the traditional "scientific" approach to problem solving cannot be applied to it. Scientific problems are tackled by gathering data, analyzing the data, formulating a solution and implementing the solution. With a wicked problem the understanding of the problem is evolving as one works 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. Ontology engineering can be described as a wicket problem.
Table 2.2 surveys the terminology used in the IBIS model.

The IBIS model is a very general and established theory to model argumentation processes and therefore the standard to describe arguments. It is, however, due to its generality difficult to apply to domain specific argumentation processes and should be extended with domain specific argument types (Potts & Bruns, 1988). The rhetorical structure theory is an argumentation model offering more fine-grained analysis methods, but it does not explain the argumentation process as IBIS does.

2.3.3. Rhetorical Structure Theory

The aim of rhetorical structure theory (RST) is to offer an explanation of the coherence of texts (Mann & Thompson, 1988; Mann, 2005). It is assumed that for every part of a co-

Term	Description
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

Table 2.2.: IBIS Terminology

herent text there is some function. RST focuses on showing an evident role for every part of a text. The RST analysis divides a text into its building blocks. According to the RST the smallest unit of text is called a span. Spans are connected through relations building a structure. The most frequent structure contains two spans of text which are virtually adjacent. These are usually related such that: 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. A very complete list of relations can be found on Mann's Web site (*cf.* (Mann, 2005)). For instance RST defines presentational relations, such as **background** that increases the ability of the reader to comprehend an element in N, and **evidence**, where the reader comprehension of S increases his belief of N. RST defines also 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. 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. Some of the relations are indicated in form of signalling words. The conjunction *"when"* for example indicates a *Circumstance*; *"but"* indicates an *Antithesis*. All relations can occur without any signalling words although for each relation one can be found. Therefore, the manual analysis of a text is very time consuming. 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 analysis 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 most important relations found in RST are: Elaboration, Evaluation, Justification, Contrast, Alternative, Example, Counter Example, Background knowledge, Motivation, Summary, Solutionhood, Restatement, Purpose, Condition, Preparation, Circumstance, Result, Enablement and List.

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
> *e.g.*:
> I suggest *knowledge management (KM)* as super concept of `DKM` because `every DKM` is a kind of *KM* Elaboration, Justification

Marcu (1997) presents an algorithm, which is able to automatically extract the relations from natural language text. The algorithm uses signaling words, but cannot extract relations with the same accuracy as humans.

2.4. Peer-to-Peer Systems

> *"The term 'peer-to-peer' (P2P) refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner."*
>
> *(Milojicic et al., 2002)*

There exist a number of definitions for peer-to-peer (P2P) systems (*cf. e.g.*, (Schollmeier, 2001; Fox, 2001; Oram, 2001)). The distinction is made between "Pure" P2P systems, "hybrid" P2P systems and "Client/Server" systems. A pure P2P system refers to network architectures in which no node in the network has a special function. This implies that any of the nodes may fail without major consequences for the entire system. In a hybrid P2P system some nodes take special responsibility for the organization of the network, while in the classical client-server model the server is responsible for the network organization. This categorization can be extended to logical layers above the underlying network layer. A logical pure P2P network may depend on a different architecture on deeper network layers[4], *e.g.*, IP routing follows a hybrid architecture.

The kind of shared resources is a distinctive feature orthogonal to the network architecture, *i.e.*, the P2P model enables (i) communication and collaboration, (ii) data sharing and (iii) distributed computing. Popular examples for communication and collaboration are Skype (Skype Limited, 2006, communication), Groove (Groove Networks, 2001, collaboration), Kazaa (Sharman Networks, 2006, file sharing) and Seti@Home (Anderson et al., 2002, distributed computing). In the following we concentrate on challenges related to data sharing applications which are in line with the ones for knowledge sharing applications.

[4]Deeper network layer refers to the categorization provided in the ISO/OSI model (Zimmermann, 1980).

2.4.1. Challenges for Peer-to-Peer Data Sharing Applications

Daswani et al. (2003) have identified a number of challenges for P2P systems. They divided them into two problem fields: search and security. A search solution proposes the *topology* of the network, the *data placement* strategy, and the *message routing* algorithm. The topology defines the arrangement of the nodes within the network and their interconnections. The data placement strategy defines were the data is located in the network and on which nodes. The message routing algorithm defines the strategy to find the desired node or information given the topology and data placement.

The definition of the three interdependent dimensions depends on the requirements of the underlying application. The requirements can be categorized as follows:

Expressiveness The the query language should be able to describe the desired data with sufficient detail. A routing mechanism supporting key lookup cannot support structured queries for relational data for example.

Comprehensiveness The comprehensiveness of the search result determines the number of returned results. Some solutions support the retrieval of only one answer to the query while others are tailored for the retrieval of all answers.

Autonomy A search solution defines the topology, the data placement and the message routing procedure. All three influence the autonomy of the peer, as the topology may predefine the neighborhood of a peer or the data placement strategy requires a specific distribution of index information across the network.

The concrete requirements affect the performance of the search solution w.r.t. the following measures:

Efficiency Efficiency refers to the absolute resources consumed required to run the search solutions. Efficiency can be measured in terms of consumed bandwidth, processing power or storage consumption.

Quality of service The provided quality of service can be measured in terms of the number of results found or the response time. Quality of service measures are user inspired, while efficiency criteria are technically inspired.

Robustness Robustness determines the search solutions behavior when peers leave and join the network. A robust search solution maintains its efficiency and quality of service in the event of failing nodes.

The authors conjecture that their exist trade-offs between the different requirements. A routing solution which requires less autonomy may offer a higher quality of service and more expressivity may come at the price of less efficiency.

The second problem field in P2P systems is related to security. Here, the following four dimensions are relevant:

Availability Attacks on the P2P network infrastructure can influence the availability of content in the P2P network. A search solutions can guarantee different service levels w.r.t. the availability of content in such events.

Information authenticity Information authenticity requires that the answer sent as a response to a query is the answer the querying peer receives. Related to this problem is the question whether the answer is the correct answer to the query or not, *i.e.*, if we can trust the responder.

Anonymity Some application scenarios require the maintenance of requester and responder anonymity.

Access control Access control enables the users to specify who can access which content within the network.

In this thesis we scrutinize the problem of search in P2P systems and focus on the problem of selecting the right peers to send a query to, thus the routing problem.

2.4.2. Routing in Peer-to-Peer Systems

> *"Routing is a means of selecting paths in a computer network along which information should be sent."*
>
> *(Wikipedia, 2006a)*

2.4.2.1. Terminology

Routing in P2P networks has attracted a lot of attention in recent years and different terminologies have been devised. In order to compare our approach to routing to other approaches we adhere to the terminology established in Cooper & Garcia-Molina (2004). They generalize the organization of different P2P systems, which use **overlays** to route queries. Overlay based routing in contrast to network based routing uses content related information to find peers. The overlay network is a logical layer on top of a physical computer network (typically IP). A **semantic overlay layer** uses semantic information to organize the network topology (Crespo & Garcia-Molina, 2002b). The **neighborhood** of a peer on the **network layer** is determined by the physical location of the peer while the neighborhood of a peer in the overlay is determined by the overlay topology. Each peer in an overlay network maintains an index of the content it shares, *e.g.*, an inverted keyword list if the searchable content are documents, or an ontology if the content is formalized as in our case. A **search** can be performed w.r.t. the information contained in the index. The index or parts of it is distributed in the P2P network according to the overlay topology, thus peers send queries to remote peers based on the index information they have obtained or received. The number of indexes a peer stores and the size of each index is restricted

by the **index size**. Peer *A* receiving index information from peer *B*, thus, establishes an **indexing link** with peer *B*. Peer *A* can create the indexing link actively by sending the indexing information to peer *B*, or peer *B* can create the indexing link passively by listening to queries or answers from peer *A*. Peer *B* not necessarily has any link with peer *A*. There can be either **forwarding** or non-forwarding indexing links. In case of forwarding indexing links peer *A* will forward the index or updates to it to its overlay neighbors, while in non-forwarding indexing links are not forwarded.
Peer *B* establishes a **search link** with peer *A* in order to send or forward a search message to peer *A*. The search link can be forwarding or non-forwarding depending if the search message should be forwarded to the neighborhood of peer *A* or not.
The number of times a message is forwarded, *viz.* the number of **hops** it travels, can be restricted by the message time to live (TTL). As a message is forwarded in the network it can be sent to peers which had already received it albeit on a different query path. To avoid those cycles query messages carry unique identifiers. The search request and its respective search message is also called **query message**. Accordingly the answer to the search request is transmitted in an **answer message**. The sequence of peers which are visited by one query message are referred to as **query path**.
Peers join and leave the network, if they do so very often this results in a high **churn rate**.

2.4.2.2. Categorization of Routing Approaches

In Section 2.4.2.1 we outline that the overlay layer determines the way indexes are distributed on the P2P network and the followed search strategies. Androutsellis-Theotokis & Spinellis (2004) introduces two dimensions in order to describe different overlay architectures: *centralization* and *structure*.

Centralization The index describing the content of the local peers can be sent to different remote peers in order to facilitate routing. This depends on the degree of centralization. In *purely*[5] *decentralized architectures* none of the participating peers takes a special role within the network, thus any peer can store index information. This has the advantage that no single point of failure is introduced to the system. In *partially centralized architectures* some well defined super nodes take special responsibility and store index information. The emerging super nodes are not necessarily a point of failure as techniques exist to replace super nodes dynamically. In *Client/Server architectures* also referred to as hybrid decentralized architectures a central peer takes all the responsibility to store indexing information and facilitate the search for relevant peers. This comes at the price of a single point of failure.

Structure The structure of the overlay network can either be build up nondeterministically or based on rules. In the first, unstructured case, the index information is either not

[5] In the next paragraphs words sans serif refer to Figure 2.3.

distributed or it is distributed without aiming at a global structure. The overlay structure is not influenced by the position of the content. In this cases brute force methods or heuristic based methods, such as flooding and informed flooding are necessary to find the requested information.
Flooding refers to routing approaches in which a message is broadcasted to all or a subset of peers which are known from the default network. A peer builds up no indexing links to any peer in the network. This approach is very message intensive and results in low quality results. The naive flooding approach can be improved if peers store some information about remote peers, without building a deterministic topology. Peers can either distribute their indexing information actively or collect it passively form remote peers. If they distribute the indexing information actively, they send it to the remote peers they are connected to on the network layer. Receiving peers may then decide to forward this indexing information to remote peers they are connected to on the network layer or not to forward it. The distribution of active indexing links results in a message overhead, each time a peer joins the network. In the passive case remote peers build up indexing information about remote peers from answers to queries. A peer establishes passive forwarding indexing links if a query response is returned to the requester on the same message path as the original query was forwarded. In the non-forwarding only the requester establishes passive indexing links, because the response is directly returned. Routing based on passive indexing links has the advantage that no overhead messages are produced if a peer joins the network, but take longer than active approaches to make informed routing discission.

In the second, structured approach to routing, the location of index information is calculated in a deterministic way. Structured P2P networks have always a purely decentralized architecture and they are designed in order to overcome the scalability problems of unstructured networks.
In order to obtain the deterministic structure a key is calculated from each value in the index. In most distributed hash table (DHT) approaches the keys are obtained using consistent hashing functions. The keys are distributed according to a predefined topology, such as a tree, a ring, a hypercup or a combination of trees and rings (hybrid). Retrieving information is straightforward in structured P2P networks using greedy search strategies, which forward request with each hop closer to the peer responsible for it; this is achieved in $\mathcal{O}(log(n))$ messages, n being the number of peers in the network. The responsible peer responses with a link to the requested information. A disadvantage of structured P2P networks originates from the fact that the index information is published each time a peer enters the network, thus producing a certain message overhead. Furthermore it is difficult to find appropriate peers for conjunctives queries as different peers can be responsible for the different parts of the query.

In Figure 2.3 we compared the routing algorithms discussed in the related work section (*cf.* Section 8.3, page 217) w.r.t. to their degree of centralization, structure and indexing strategy.

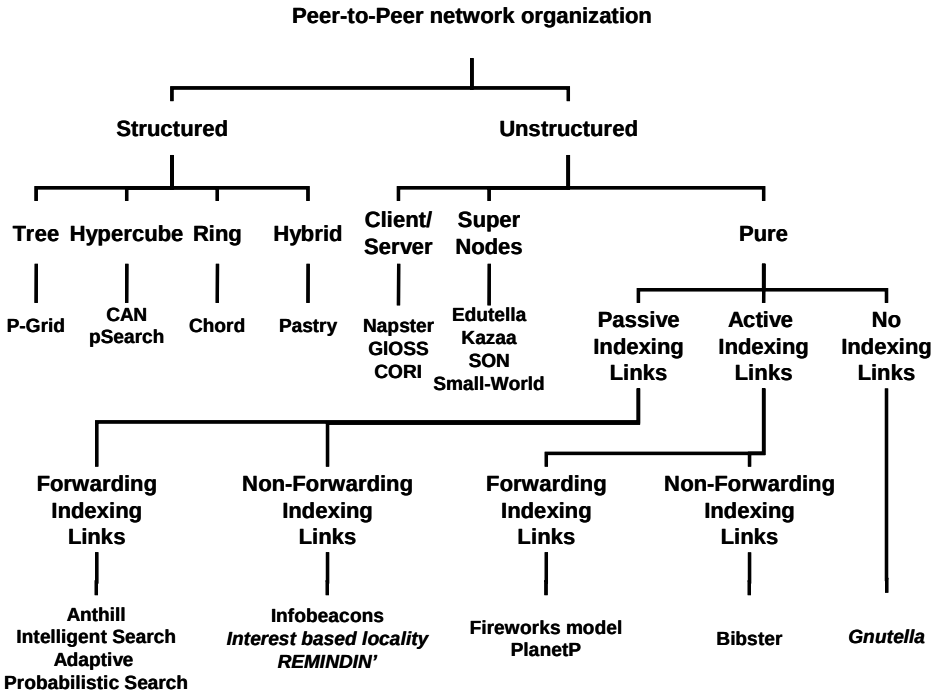


Figure 2.3.: Categorization of Routing Approaches

3. SWAP: A Semantic Peer-to-Peer System

Overview

In this chapter we present the SWAP system. The SWAP system offers basic services to exchange knowledge in a peer-to-peer fashion. We derive the generic requirements for this infrastructure from use cases carried out in the SWAP project. We propose a modular system architecture which allows for the customization of the generic platform to specific user needs. The components extract knowledge from the user machine, visualize the knowledge, and connect to other nodes. A metadata model enables knowledge sharing across the network.

References: This chapter is based on the publications (Tempich et al., 2003), (Broekstra et al., 2003) and (Tempich et al., 2004c).

3.1. Use Case Description

The trend that knowledge workers are collaborating in decentralized, autonomous teams is reflected in the two use cases which lead to the requirements definition for the generic SWAP system. The first use case describes a virtual organization in the tourism domain, while the second describes a community of researchers exchanging bibliographic metadata. The use cases are representative for scenarios in which a number of organizations/people desire to share knowledge on particular but changing topics while maintaining their autonomy.

3.1.1. The IBIT Use Case

The IBIT case study is based in the tourism sector of the Balearic Islands (Lladó et al., 2002). The needs of the tourism industry there are best described by the term 'coopetition'. On the one hand the different organizations *compete* for customers against each other. On the other hand, they should *cooperate* in order to provide high quality for regional touristic issues like infrastructure, facilities, clean environment, or safety — that are critical for them to be able to compete against other tourism destinations. The different organization thus form a virtual organization (*cf.* (Camarinha-Matos & Afsarmanesh, 2003)). Therefore, they collect and share information about *indicators* reflecting the impact of growing population and tourist fluxes in the islands, their environment and their

infrastructures. Moreover, these indicators can be used to make predictions and help planning. For instance, organizations that require *Quality & Hospitality management* use the information to better plan, for example, their marketing campaigns. As another example, a governmental agency, a Balearic government co-ordination center of telematics, provides the local industry with information about *new technologies* that can help the tourism industry to better perform their tasks.

The employees prepare different reports on their computers. They deliver them to the government and to interested parties on request. Some of the produced reports are public while others have a restricted to specific audiences. The organizations produce periodical reports, such as statistical ones, and reports on upcoming technological developments. The data is collected via email, from Web sites and non electronic sources. Locally, the source data is organized in emails, documents, tables, and different folder hierarchies for files, bookmarks and emails. Although the gathering of information and its distribution to interested parties is time consuming, none of the organizations has enough resources or expertise to set-up and maintain a centralized knowledge base for publishing knowledge of general interest. Security concerns also impede the introduction of such a system.

3.1.2. The Bibster Use Case

The SWAP bibliography case study explored the sharing of BibTEX information between peers of researchers. BibTEX is a metadata standard to describe bibliographic information. As researchers reference related work to their own research they collect this data. The collection of bibliographic information is time consuming and it is sometimes difficult to obtain all required information, *e.g.*, the pages, for a particular reference. Although central repositories[1] exist they usually do not include bibliographic information across different domains and cover only larger conferences and journals. Bibliographic information of smaller events resides only at the individual researcher. Beyond the retrieval of bibliographic information the comparison of different researcher data can on the one hand assist to complete bibliographic data and on the other hand can reveal similar research interests. The categorization of bibliographic data is feasible, since for many research areas shared topic definitions exists, such as the ACM topic hierarchy for computer science[2].

3.2. Requirements for a Semantic Peer-to-Peer System

The use cases presented in the previous section are DKM scenarios. In order to formulate the specific requirements for a semantic P2P system supporting the use cases we categorize them according to the structure presented in Section 2.1.2. For the requirements related to search and security we use the more detailed categorization presented in Section 2.4.1.

[1] For example: DBLP, see `http://www.informatik.uni-trier.de/~ley/db/` or The Collection of Computer Science Bibliographies, see `http://liinwww.ira.uka.de/bibliography/`

[2] see `http://www.acm.org/class/1998/`

3.2.1. Infrastructure Level

On the infrastructure level we define requirements related to access and security.

Access The SWAP system enables the exchange of knowledge represented in RDF(S). Furthermore, the use cases require the exchange of, *e.g.*, documents. Documents existing in different versions should be distinguishable. The system transmits both kinds of data across organizational boundaries, such as fire walls.

Security For our use cases we do not expect a denial of service attack or malicious nodes. Users provide only usable and trustable knowledge. Furthermore, they do not pretend to author files which they have not. Hence, for our case studies we need no file authenticity validation service. Neither, do the users necessitate anonymity. In the Virtual Organization case study access control is a requirement. Users want to grant access to their files only to certain organizations, and on special request. Although we neglect some general security requirements this does not undermine the value of the system, as security services can be implemented on top of the generic infrastructure.

3.2.2. Application Level

The main objective of the SWAP system on the application level is to support the knowledge management core processes identification, acquirement, distribution and use (*cf.* Figure 2.1).

Local perspective The knowledge workers can create their own view on the local and remote knowledge. They create local views comparable to the structures found in their file system. The local perspective changes over time in relation to the differences in knowledge workers tasks, but they also construct shared perspectives for knowledge relevant for the team. Mappings may be required, *e.g.*, to overcome heterogeneous labeling of the same objects. If the defined structures are very similar to each other, a process is needed to identify commonalities and make them explicit.

Owner ship model An owner ship model is not seen as a major requirement for our case studies.

Content-based search In both case studies it is required to search content related. The knowledge is categorized according to different dimensions. In some cases only imprecise search queries can be formulated. The visualization of the answers should point up the similarities between different queries. Furthermore, the visualization should account for the fact, that answers in a P2P network arrive at different times. Hence, the interface should

help the user to distinguish between recent and old results, should update itself from time to time and should visualize where results come from.

Localization In the case of localization the requirements are subdivided according the categorization provided in Section 2.4.1. Search is performed on the content level, thus an expressive query language is required. It is not sufficient to return only one answer, but a subset of all possible answers to a query. As the search request may not completely cover the information need, a larger answer set can contain the desired information. Peers are autonomous and like to connect to any remote peer available in the network, as they may know of somebody who knows the answer to their question.

The resources peers can allocate to the system are limited, as the retrieval of knowledge is not their core task but supportive. Regarding quality of service, users expect quick answers to their queries. The system should cope with peers joining and leaving the network. Although in the IBIT case study we can assume that peers are most of the time online during a working day.

3.2.3. Community Level

Formation In both case studies participants appreciate if they are introduced or pointed to peers, which have similar knowledge as they have. Similarity can be determined on the content level. Peers also connect to remote peers, if they know them.

Community perspective The creation of a community perspective on the shared knowledge is of particular concern in the IBIT case study. The participants need a process guiding them in the construction of shared knowledge models. In the Bibster case study, the ACM hierarchy is predefined.

Interoperability In the Bibster case study interoperability is guaranteed, as all peers stick to the BibTEX metadata format and use the ACM topic hierarchy. In the IBIT case study existing shared knowledge structures can facilitate interoperability, but people also create individual views on their knowledge. In order to exchange knowledge these peers should create associations between the different knowledge representations.

Free riding In our use cases free riding is of no concern.

Summary

In comparison to general purpose data sharing P2P networks and traditional knowledge management (KM) systems the DKM setting has the following requirements: A DKM

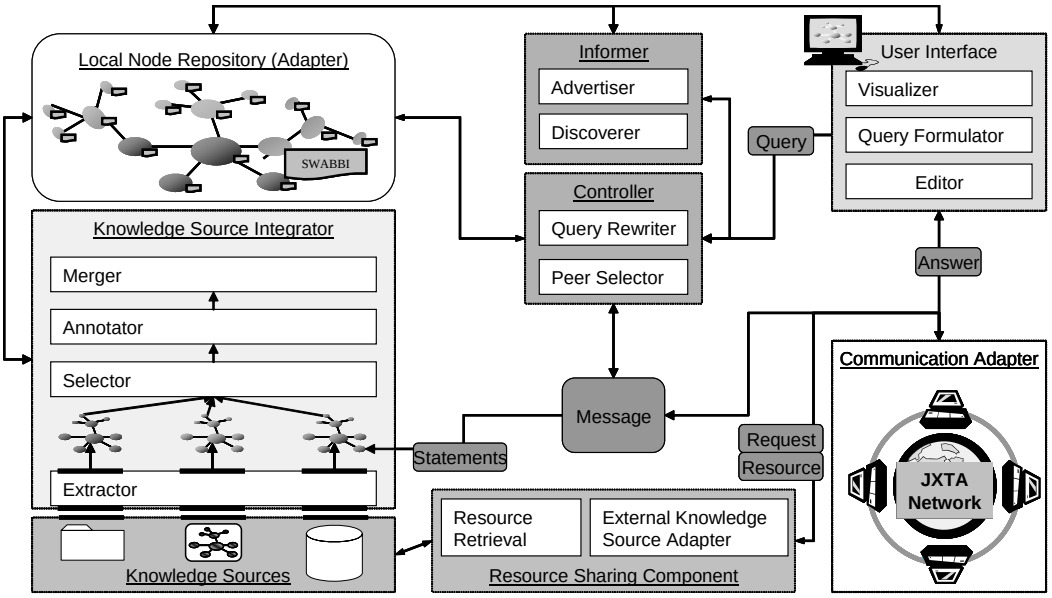


Figure 3.1.: Abstract Architecture of a SWAP Node

system should support knowledge sharing opposed to data sharing and representation of user knowledge on local machines opposed to the representation of centrally predefined knowledge. The represented knowledge is sensitive and cannot be shared with everybody. The knowledge model evolves quickly and differently at each peer. The peers are not always available.

3.3. A Generic Semantic Peer-to-Peer System Architecture

In order to meet the requirements listed in the previous section we have devised the SWAP system architecture depicted in Figure 3.1. We will now present the individual components.

Local Node Repository The Local Node Repository (LNR) stores the knowledge of a peer and the metadata required for mediation between views, peer selection and resource location. It provides query formulation and processing functionalities. Knowledge is represented as RDF(S) statements in the LNR. The LNR is implemented building on the Sesame RDF(S) repository and SeRQL query language (Broekstra et al., 2002; Broekstra, 2003). The LNR provides an integrated view on the knowledge represented in RDF(S) but also on the documents from which the knowledge was obtained. The native Sesame engine is integrated with an document retrieval system in order to tradeoff Sesame's performance w.r.t. queries for statements and the retrieval system perfor-

mance for keyword word based search (Plechawski, 2004b). It thus meets the requirement *Content-based search* and is the basis to meet the requirement *Access* in combination with the Knowledge Source Integrator, the Communication Adapter and the Resource Sharing Component.

The SWAP knowledge model ontology and the SWAP metadata ontology are part of each LNR. The SWAP knowledge model presented in Section 3.4.2 conceptualizes metadata about the knowledge sources being shared in the network. The SWAP metadata model described in Section 3.4.1 represents information to perform system related tasks.

In order to meet the *Security* requirements the SWAP system uses a public-key infrastructure (PKI) with certificate authorities (Plechawski, 2004a). One SWAP node acts as a root certificate authority for the SWAP system, all other peers will configure this node as trusted root certificate authority. In small networks, this certificate authority will issue certificates directly for users, whereas in large networks, it is possible to build a hierarchy of certificate authorities. The certificate creation is done offline, on the configuration level. Certificates themselves are not transmitted within the standard SWAP user interface.

Knowledge Source Integrator The Knowledge Source Integrator consists of the four sub components Extractor, Selector, Annotator and Merger. It is responsible for the extraction and integration of internal and external knowledge sources into the LNR. This task comprises (1) means to access local knowledge sources and extract an RDF(S) representation of the stored knowledge, (2) the selection of the RDF statements to be integrated into the LNR, (3) the annotation of the statements with metadata, and (4) merging the statements into the ontology of the user.

The Extractor offers a common interface to access different kinds of data on the local machine. It follows the mediator design principle (Wiederhold, 1992). For each information source, such as a file system, an extractor is implemented, which can extract the information modeled in the SWAP knowledge model (*cf.* Section 3.4.2). The Extractor produces and RDF(S) representation of the extracted information. The Selector choose from the extracted statements the relevant ones. In our implementation we have always integrated all the extracted information. The Annotator adds metadata information to the selected information according to SWAP metadata model. The Merger integrates the annotated statements into the existing LNR. Different strategies are available to support the merging process. Ehrig (2006) describes them extensively and shows how they meet the *Interoperability* requirement.

Knowledge Sources Peers have local sources of knowledge such as the local file system, email directories or bookmark lists. These are the basis to create the knowledge of a peer as well as its basic vocabulary. The peer may also share the knowledge sources.

Informer The Informer consist of two subcomponents: the Advertiser and the Discoverer. It publishes and searchers for expertise descriptions in the P2P network and thus

builds active indexing links with remote peers (*cf.* Section 2.4.2.2). The Informer extracts from the LNR a model of the peer expertise. The Advertiser transmit this model to remote peers whereas the Discoverer requests the model. In both cases the receiving peer decides to store or to disregard the expertise model. The expertise models of remote peers assist in the peer selection process. A more detailed description of the advertisement and discovery service can be found in (Haase et al., 2004b). The routing approach proposed in this thesis does not require the Informer to localize knowledge. The Informer in combination with the Peer Selector meets the *Localization* requirement.

Controller The Controller contains the Peer Selector. It is the system coordinating component which controls the message flow within the system and administrates the query distribution and answering process. The Peer Selector picks from known remote peers appropriate ones, considering the peer local knowledge and the query. Part III proposes algorithms implemented in the Peer Selector component to select remote peers. The Peer Selector meets the requirement *Formation*, as the proposed algorithms clusters peers with similar interests, and a manual Peer Selector allows for selecting arbitrary peers to query.

Resource Sharing Component The Resource Sharing Component delivers resources to a requester which are not part of the LNR.

User Interface The User Interface has the subcomponents Visualizer, Query Formulator and Editor. The Visualizer presents the answers to the user queries. The XAROP user interface presents the answers to queries in a cluster map (*cf.* Figure 4.9). It presents the results of more than one query at a time, and points out overlapping results. It updates itself when new answers arrive from remote peers and highlights them (*cf.* requirement *content-based search*). The Bibster user interface presents the answers to queries in table format. The Query Formulator is similarly represented in both prototypes. It allows to query for instances by selecting classes from a class hierarchy. Properties and keywords can be used to specify the query further. The Editor supports the creation, maintenance and management of the local knowledge. It assists the creation of a *local perspective* on the user knowledge as well as the creation of a *community perspective*. The process structuring the perspective creation is described in more detail in Part II whereas Section 4.5.2 emphasizes on the Editor component.

Communication Adapter The Communication Adapter is responsible for the network communication between peers. It serves as a transport layer for other parts of the system, for sending and forwarding queries. It hides and encapsulates all low-level communication details from the rest of the system. The JXTA protocol provides communication services to surpass fire-walls and to find remote peers on the network layer (Gong, 2001).

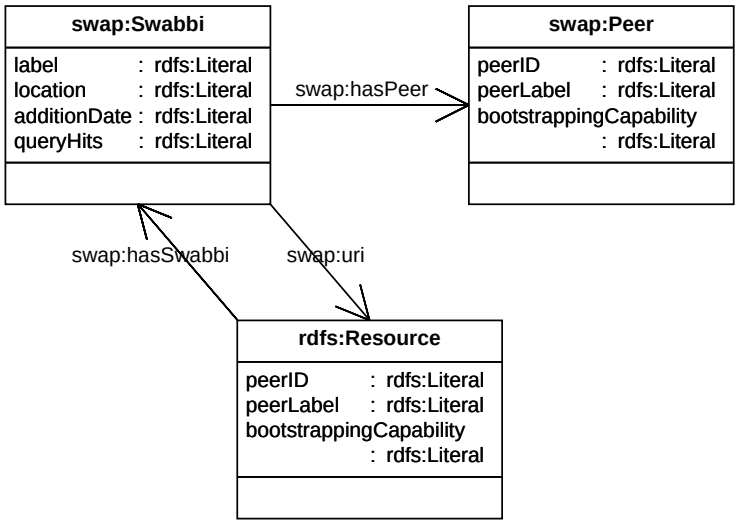


Figure 3.2.: SWAP Metadata Model

3.4. SWAP Application Metadata

In order to meet the requirements *Access* and *Localization* the LNR integrates two predefined metadata models. The SWAP metadata model specifies information required to localize data within the network, while the SWAP knowledge model describes the information found on the local peer. All metadata is integrated into the LNR.

3.4.1. SWAP Metadata Model

The LNR integrates knowledge from different knowledge sources locally and remotely. In order to trace the origins of knowledge objects the Annotator adds metadata according to the SWAP metadata model to it. This applies to the physical location on the local machine in the same way as to the physical location at a remote peer. The *Access* requirement motivates the metadata related to the location of specific knowledge objects. Metadata related to the quality of the knowledge objects arises from the *Localization* requirement.

The metadata is represented in RDF(S)[3]. Figure 3.2 visualizes the defined classes.

The model consists of two RDFS classes, namely the “Swabbi”-class and the “Peer”-class. The following properties specify the two classes:

Swabbi-object Every object in the LNR is annotated with a “Swabbi”-object containing the following meta-information:

[3] see `http://swap.semanticweb.org/2003/01/swap-peer`

hasPeer This property is used to track which peer this "Swabbi"-object is associated with.

uri Each object, *i.e.*, instance, class or property was originally created on a specific peer. To store the knowledge origin and to disambiguate object addresses across the network its URI is explicitly stored.

location Whereas the URI identifies metadata resources within the ontology of the LNR, the location-attribute is an identifier to access the physical resource, *e.g.*, a document at `file://c:/Projects/myfile.txt`. The peer storing the information physically interprets this expression. The location information is also required for updating the LNR.

label The label attribute stores the name of an object at its origin. The label-attribute is expressed in natural language. A resource can have different names on different peers. This information is useful for mapping purposes.

queryHits The queryHits attribute is used in the Peer Selector to rank peers. The maintenance of the associated values is described in Section 6.3.

additionDate This attribute stores the addition date of the associated object. The Peer Selector indexing policy is influenced by the addition date of an object.

Peer-object The Peer-object is an RDFS representation of a node in the network. Each Swabbi-object is related to a Peer-object by the hasPeer relation. The following properties specify the Peer-object:

peerID Each peer has a unique ID. For our purposes this will be the *JXTA UID*, as we use JXTA as our underlying communication infrastructure. The ID provided by JXTA is a random number with a high probability being indeed unique.

peerLabel This peer attribute stores the peer label, which is a human readable and understandable description of the peer in natural language.

bootstrappingCapability The bootstrapping capability is indicator for the number of connections a peer maintains. The Peer Selector incorporates the bootstrapping capability in the peer selection decision.

3.4.2. The SWAP Knowledge Model

The SWAP knowledge model represents the information commonly found on a user machine. It is represented in RDF(S)[4]. It contains classes for, *e.g.*, Folder, File, Bookmark and Email. Folders are arranged in a hierarchy and specializations of the concept exist for, *e.g.*,

[4] see `http://swap.semanticweb.org/2003/01/swap-common/`

FileFolder and BookmarkFolder. The knowledge model specifies for each class a number of properties. The properties author, size, checksum and inFolder for instances specify the class File. Without presenting all classes defined in the knowledge model we conclude that it facilitates the retrieval of information found on each peer. The information extracted according to the knowledge model is the basis for the creation of knowledge from local information.

3.5. Summary

This chapter introduces two use cases for distributed knowledge management. In the first use case virtual organizations exchanges knowledge across organizational boundaries in order to provide better services to their customers. In the second use case researchers exchange bibliographic entries easing the burden to collect complete references and to detect colleges with similar interests. To meet the requirements of the use cases from a technical point of view the SWAP system has been developed. The SWAP system is a generic peer-to-peer platform offering services to extract, store and maintain knowledge stored on the local node. It also enables the exchange of knowledge between nodes in the network, thereby integrating ontologies from various sources. This provides a basic infrastructure for distributed ontology engineering and mapping of ontologies. The REMINDIN' routing algorithm extensively uses the shared ontology to efficiently route queries in the network. In comparison to related system it uniquely offers to query arbitrary structured data stored in RDF(S) and unstructured data such as text at the same time. The SWAP system has been customized for the two use cases in the XAROP (virtual organizations) and Bibster (researchers) application. This demonstrates one of the main benefits in using a standardized data model: the system may be customized for different application scenarios in a straightforward manner. Moreover, the XAROP application includes a completely distributed security model as required in DKM settings.

The SWAP system was also used to implement the distributed ontology management infrastructure Oyster (Hartmann et al., 2005). It is currently enhanced in order to support distributed reasoning with OWL ontologies.

Part II.

The DILIGENT Methodology

"Strategy without tactics is the slowest route to victory.
Tactics without strategy is the noise before defeat."
— Sun Tzu

4. DILIGENT Ontology Engineering

DILIGENT is a methodology to build, evolve and reconcile ontologies in a DKM settings. It defines a process model, guiding the methodology users towards a shared ontology in a structured way, characterizing roles, specifying the required knowledge of the users. The methodology is supported by tools, assisting the users in particular application scenarios. The methodology has been successfully evaluated in case studies.

Overview

In this chapter the general process overview and role characterization is followed by a detailed discussion of the process stages and their activities. As arguments play a predominant role in the methodology, ontology engineering is analyzed from an argumentation point of view and a semi-formal model for argumentation support is presented. The description of tools supporting the methodology user in different application scenarios finalizes the chapter. The evaluation of the methodology in three different case studies is presented in the next chapter.

References: This chapter is based on the publications (Pinto et al., 2004a), (Pinto et al., 2004b), (Pinto et al., 2005), (Vrandecic et al., 2005) (Tempich et al., 2005a), (Tempich et al., 2006), and (Sure et al., 2004).

4.1. Feasibility Study

This Section elaborates on the IBIT use case introduced in Section 3.1.1 w.r.t. its requirements on ontology engineering methodologies. Existing ontology engineering methodologies cover these requirements only partially. The DILIGENT methodology presented in the next Section addresses the open issues.

4.1.1. Ontology Engineering Use Case

We focus on the IBIT use case here, because the Bibster case study did not foresee changing the common model.

The organizations in the IBIT use case have the shared interest to exchange knowledge about topics which are of common interest, *e.g.*, visitor numbers, or are orthogonal to their core competencies, *e.g.*, hotel booking systems. The exchanged knowledge is modeled with an ontology, because they require the explicit semantics of a formal knowledge model. In contrast to the centralized KM setting the domain experts build the ontology themselves, because no specialist ontology engineer is in charge of maintaining the local ontologies. A major advantage of DKM is its responsiveness to changing needs; the complete delegation of the ontology maintenance task to a centralized authority would challenge this advantage. The ontology reflects the changing needs in that it is changed by each user differently. Nonetheless, some of the changes made by one user are similar to other user changes.
Another important aspect of the IBIT use case is the fact that new employees enter participating organizations and others leave them. There is thus the potential of reusing existing ontologies in order to ease the knowledge transfer between employees.

4.1.2. Requirements for Ontology Engineering Methodologies

We categorize the requirements for our ontology engineering (OE) methodology into general requirements for a methodology, requirements for OE methodologies and requirements for OE methodologies in our scenario.

A methodology comprises the description and definition of methods, techniques, processes and activities which are designed to achieve a certain goal (*cf.* Section 2.2.2). Besides the process itself a methodology includes also role definitions for the actors accomplishing the activities as well as the input and output documents and results of the activities. Early requirements analysis for OE methodologies concluded, that such a methodology should include activity descriptions for *Purpose identification*, *Ontology building*, *Ontology Evaluation* and *Documentation* (Uschold & King, 1995). An extended list includes all ontology management, ontology development oriented and ontology support activities (*cf.* Section 2.2.2.2, (Corcho et al., 2003)). A methodology provides these activity descriptions suitable for a particular class of use cases. Therefore, we focus on the requirements arising from our use case and compare them with the assumptions underlying existing methodologies.

Decentralization Classical development of ontologies is mostly *centralized* like the targeted knowledge-based system where ontologies are used. Existing methodologies, have a centralized approach towards engineering knowledge structures requiring *knowledge engineers*, *domain experts* and others to perform various tasks, such as *requirement analysis* and *interviews*. While the user group of such an ontology may be large, the development itself is performed by a comparatively small group of domain experts who *represent* the user community and ontology engineers who *help structuring*.

In contrast, in our scenario the *users* build the ontology in a decentral and autonomous manner. Moving the user in the role of the ontology engineer implies a number of requirements elaborated on hereafter. This applies also to autonomy. In a decentralized setting the communication of ontology design decisions to other users is of particular concern. Although an ontology explicitly specifies the meaning of its concepts, not all assumptions underlying a particular formalization are representable (Skuce, 1995). Since, the users do not meet each other in order to explain the assumptions they require methods to facilitate their provision.

Non Expert Builders Existing methodologies support KE by using check lists that guide the engineering process. The check lists have been shaped by the needs of *knowledge engineers* to comprehensively cope with nearly arbitrarily complex processes. In contrast, in the 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* and/or *users*. Since, ontology engineering is not the primary task of this group, they require fine-grained guidance and tools integrated into their work environment in order to efficiently build and change their ontology and explain the underlying design decisions.

Autonomy In contrast to the centralized scenario, the users can change their ontology according to their needs, but they have also an interest in sharing knowledge with other users in the network. They, thus, require methods and tools helping them in bridging heterogeneous conceptualizations and agreeing on shared conceptualizations. For that the communication of ontological assumptions is important.

Iteration Existing OE methodologies focus on the construction of ontologies for an *up- and running systems* with some moderate effort for maintenance. In contrast, ontologies in our scenario permanently *evolve* in order to reflect the widely diverging needs of their users. This requires that the methods and tools defined in order to meet the before mentioned requirements should inherently consider iteration.

Summary

In DKM scenarios the users move in the center of attention. As they are non experts in ontology building they need a fine-grained guidance in order to change their ontology according to their needs. The exchange of knowledge, given the emerging heterogeneity, is only feasible if either mappings can be defined or a process supports reconciliation. Reconciliation requires discussions and the clarification of assumptions underlying different conceptualizations. In decentralized settings, in which the user do not have regular ontology engineering face-to-face meetings, an efficient clarification process requires that the assumptions are made explicit. Additionally a methodology should account for the rapid change user ontologies undergo.

We account for these requirements in the DILIGENT methodology introduced in the next Section. The methodology guides the users in a periodic process of changing, reconciling and evolving their ontologies. A partially shared ontology emerges inspired by the user changes. The methodology meets the general requirements for an OE methodology meeting the special needs of the DKM scenario. An argumentation framework structures the reconciliation process. It integrates an argumentation model which facilitates the externalization of assumptions and makes the reconciliation process more efficient.

4.2. The DILIGENT Process

DILIGENT stands for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. It addresses the requirements for ontology engineering in DKM settings. For that it distinguishes two 'kinds' of ontologies: the **shared ontology** and the **local ontologies**. The shared ontology is available to all users but they cannot change it directly. In order to change the shared ontology the user obtains a copy of it and modifies it in his work environment. The resulting ontology is referred to as the user local ontology. In the perspective of a specific user, the local ontologies of other users are referred to as **remote local ontologies**.

In this Section we sketch the overall process model and the general idea of DILIGENT. In particular we motivate the process for building and maintaining the shared ontology while the users utilize their own local ontologies. In the next Section we provide a detailed description of the activities each process stage is comprised of.

4.2.1. Key Roles

In a DILIGENT scenario we differentiate among a number of complementarily skilled experts who collaboratively build a shared ontology. In a virtual organization they often belong to competing organizations and are geographically dispersed. DILIGENT supports trained ontology engineers as well as typical users of information systems. The ontology engineers perform the defined activities with more accuracy and awareness of the process while the non-ontology engineering expert users will follow them implicitly guided by the provided tools.

DILIGENT distinguishes between the users of the ontology and a board. The users work with their local ontologies while the board assures that the shared ontology evolves according to the user needs. The board should have a well-balanced and representative participation of the different kinds of participants involved in the process: Domain experts, ontology engineers and users. Users are involved in ontology development, through their requests and re-occurring improvements and by evaluating it, mostly from a usability point of view. Domain experts in the board are responsible for evaluating the ontology, mostly from a technical and domain point of view. Ontology engineers analyze arguments and

balance them from a technical point of view. Another task for the board is to assure some compatibility with previous versions.

In future we refer to the participant in the process as *actors* or *ontology users* independently of their personal skills in ontology engineering if the local ontology is concerned. We refer to the participants involved in building and updating the shared ontology as the *board*.

4.2.2. Process Stages

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 core ontology. This central board is also responsible for deciding to update the core ontology. However, updates are mostly based on changes re-occurring at and requests by *decentral*ly working users. Therefore, the board only *loosely controls* the process. Due to the changes introduced by the users over time and the on-going integration of changes by the board, the ontology *evolves*. At the next finer level of granularity DILIGENT comprises five main stages: (1) **build**, (2) **local adaptation**, (3) **central analysis**, (4) **central revision**, (5) **local update** (*cf.* Figure 4.1).

4.2.2.1. Central Build

The process starts by having *domain experts*, *users*, *knowledge engineers* and *ontology engineers* **build** an initial ontology. In contrast to existing ontology engineering methodologies (*cf.* (Staab et al., 2001; Gangemi et al., 1998; Gómez-Pérez et al., 2003; Pinto & Martins, 2001; Uschold & King, 1995)), we do not require the initial shared ontology to completely cover the domain of interest. 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.

4.2.2.2. Local Adaptation

Once the core ontology is available, users work with it and, in particular, adapt it to their local needs. They have their own business requirements and correspondingly evolve their local ontologies (including the common core) (Noy & Klein, 2003; Stojanovic et al., 2002). In their local environment, they are also free to change the reused core ontology. This has no direct effect on the shared ontology or local ontologies of other users. Logging local adaptations (either permanently or at control points), the control board collects change requests to the shared ontology. Depending on the application and the use case, the board may either collect the changes automatically from the users, the users send the formalized changes to the board, or the users send informal requests to the board.

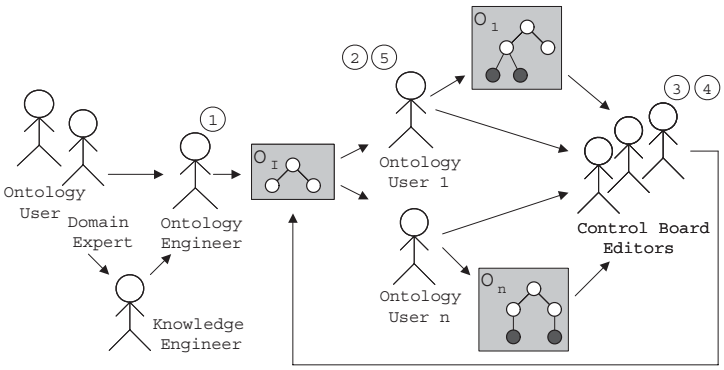


Figure 4.1.: Distributed Ontology Engineering: Overview

4.2.2.3. Central Analysis

In order to update the shared ontology in correspondence with new user requirements a board **analyzes** the local ontologies and change requests and tries to identify similarities in user ontologies. Since not all of the changes introduced or requested by the users will be introduced to the shared core ontology,[1] a crucial activity of the board is deciding which changes are 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 their evolving requirements[2] has to be found. The analysis stage concentrates on the identification of new requirements from a conceptual point of view. The changes are formalized in the next stage.

4.2.2.4. Central Revision

The board regularly **revises** the shared ontology, so that local ontologies do not diverge too far from the shared ontology. Revision can be regarded as a kind of ontology development guided by a carefully balanced subset of evolving user driven requirements. Ontology engineers are responsible for updating the ontology, based on the decisions of the board.

4.2.2.5. Local Update

Once a new version of the shared ontology is released, users may **update** their own **local** ontologies. The users tradeoff the benefits of using a shared ontology with the additional

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

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

effort to update. Some of their change requests may be part of the new shared version of the ontology. Updating may involve a reorganization of the local ontology. The individual users benefits from the use of the shared ontology, because he gains interoperability with other users. In some cases this comes at the cost of loosing the individual local organization.

4.3. DILIGENT Detailed Process Description

In order to facilitate the ontology engineering process, DILIGENT provides detailed guidance to its users. For each stage DILIGENT describes (i) major roles, (ii) input, (iii) decisions, (iv) activities (v) and output information. The result is depicted in Figure 4.2. Roles relate to the skills of the participants involved in a process stage. The input describes the available information, which may be utilized in a specific process stage. The participants perform the activities and process the input information. Based on the outcome they decide on the follow up activities and produce the desired output.

DILIGENT defines the activities at the same abstraction level as other established methodologies, which are specified for ontology engineers. The methodology, however, shall assist non-ontology engineering experts. It therefore subdivides some activities and introduces **user tasks**. The user tasks are related to the experiences in our use cases and take into account the users specific work environment.

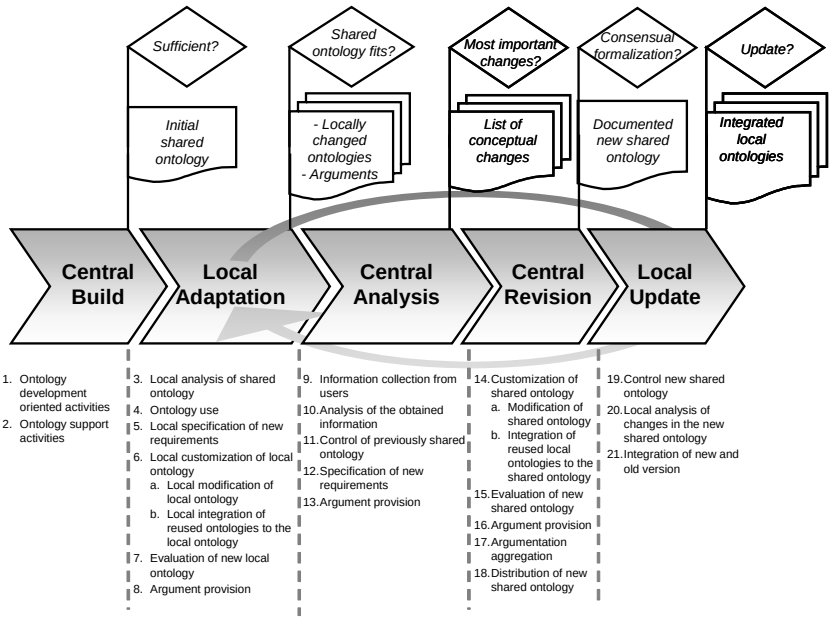


Figure 4.2.: DILIGENT Process

4.3.1. Central Build

As mentioned in Section 4.2, DILIGENT focuses on distributed ontology development and the process of ontology evolution. We do not elaborate extensively on initial building, because it is supported by established methodologies. DILIGENT, however, reuses some of the activity and structure definitions from established methodologies in later stages of the process. Therefore, we summarize those activities hereafter (*cf. e.g.*, (Gómez-Pérez et al., 2003)). Additionally, DILIGENT introduces the *Argument provision* activity, because its outcome is needed in later stages.

4.3.1.1. Roles

Classical ontology engineering methodologies introduce three different roles: knowledge engineer, ontology engineer and domain expert (as described in Section 2.2.2.2). The persons involved in the central build stage are the initial board members.

4.3.1.2. Input

If an ontology is used in an application, ontology building is integrated into the software development process. For the development of knowledge management applications the OTK methodology proposes a knowledge meta-process (Sure, 2003; Sure & Studer, 2002). In this case ontology building is preceded by a feasibility study. Besides a number of documents judging the economical, technical and organizational feasibility of the application, the study results in a detailed description of the organization for which the application is build, a list of the tasks the organization performs and an evaluation of the impact the KM application could have on the organization.

4.3.1.3. Output

The result of the central build stage is an ontology, which models the main concepts of the domain. This stage is explicitly not considered as a complete OE cycle as its objective is the realization of a first working draft of the shard ontology at the cost of completeness.

4.3.1.4. Decisions

The ontology engineer controls whether the built ontology meets the requirements identified in the beginning of the ontology engineering process. If it does, the board releases the first version of the shared ontology otherwise the building process is restarted.

4.3.1.5. Activities

In the following we summarize the activities described in existing ontology engineering methodologies for building ontologies, because later stages of the process include some of them. In particular we describe the ontology development activities *Specification*, *Conceptualization*, *Formalization*, *Implementation*, *Maintenance*, and the ontology support activities, *viz. Evaluation*, *Reuse*, *Knowledge acquisition*, *Documentation* and *Argument provision*. Figure 4.3 illustrates the interdependencies in an activity diagram.

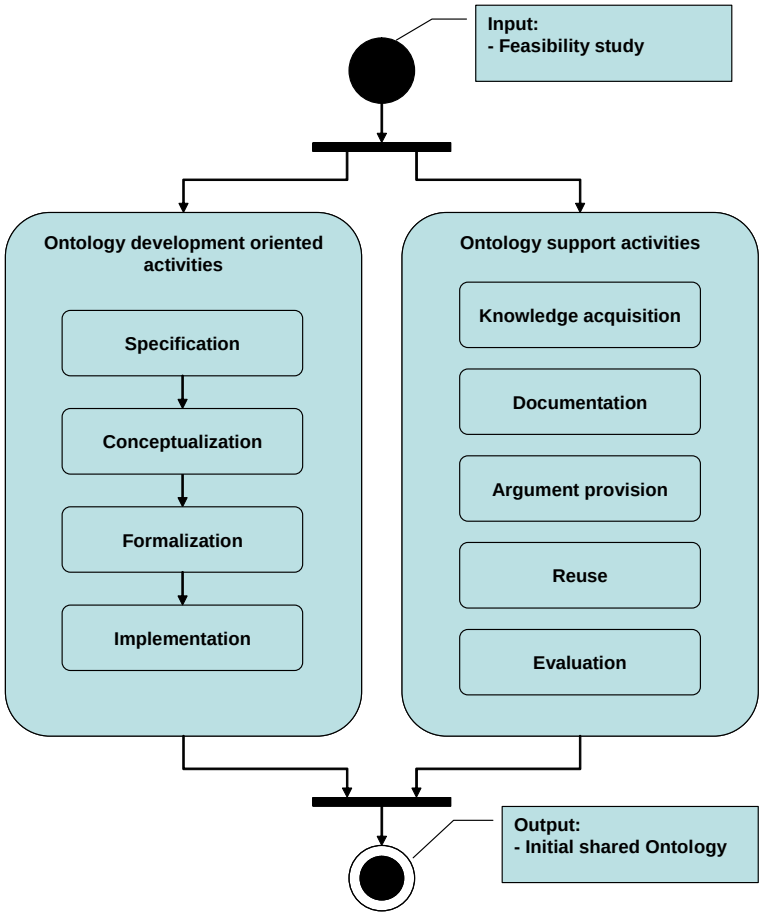


Figure 4.3.: Central Build

Specification of the shared ontology The objective of the *Specification stage* is the identification of the purpose and scope of the shared ontology. For instance, the OTK methodology situates this activity in the kickoff phase. The result of this activity is an ontology requirements specification document (ORSD) (*cf.* (Sure, 2003)). The ORSD con-

tains information about the goal, domain and scope, it specifies design guidelines and lists knowledge sources, potential users, usage scenarios and supported applications. Competency questions are often collected in order to state a set of controllable requirements; use case diagrams are another option (De Nicola et al., 2005). Supporting techniques for this activity are, *e.g.*, writing motivational scenarios or brainstorming (Uschold, 1996). Furthermore the ontology engineers define modules covering sub-domains in order to break down complexity.

Conceptualization of the shared ontology The specification defined in the previous activity is the basis for the creation of a conceptual model. The conceptual model can be defined according to different paradigms, such as the informal sketchy mind map™ model (Sure, 2003) or a semi-formal binary relations diagram together with a concept dictionary (Fernández-López et al., 1999) or others. The conceptualization complies with a predefined naming convention, elaborates the sub-concept relationships and identifies the important relations. The resulting model needs not to cover the domain knowledge completely. The ontology should represent as much detail as necessary for its initial usability and usefulness. However, the building costs for the initial ontology should not outweigh its benefits. Unfortunately this decision depends on the experience of the ontology engineer and domain expert, but attempts are underway to estimate the effort distribution between initial build stage and later stages of the process based on effort estimation models (Paslaru Bontas & Tempich, 2005).

Formalization and implementation of the shared ontology In this activity the conceptual model is converted into a formal model and subsequently implemented in the target representation language (*cf.* Section 2.2). As most building processes are supported by ontology engineering environments (OEEs) the tool implements the ontology based on the formal model. Axioms are a common way to explicitly specify the meaning of a concept.

Knowledge acquisition Knowledge acquisition has the objective to elicit relevant domain knowledge from, *e.g.*, domain experts, ontologies and literature. Knowledge acquisition has a long history and a number of techniques like brainstorming, interviews, questionnaire, text analysis, and inductive techniques have been proposed (Boose, 1989; Gaines, 1989). Besides these techniques, methodological support for knowledge acquisition has been investigated (Studer et al., 1998). As the acquisition of knowledge is time consuming, research effort has been spend alleviating the knowledge acquisition bottleneck by automatizing the knowledge acquisition process (*cf. e.g.*, (Maedche & Staab, 2001)).

Reuse This activity describes the reuse of available ontologies and their integration into the target ontology. The literature distinguishes two cases of ontology reuse processes: merging and integration (Sowa, 2000; Pinto & Martins, 2004).

Merging describes the process of building one ontology out of different source ontologies from the same domain. For that the ONIONS methodology defines a process model (Gangemi et al., 1998). **Integration** describes the process of building one ontology out of different source ontologies from different domains. For this case Pinto & Martins (2001) propose a methodology. Both methodologies start with the retrieval, selection and analysis of existing, reusable ontologies. The identification of the assumptions underlying the reused ontologies is a major task. If the ontologies are merged the methodologies suggest defining first an ontology at a higher level of abstraction than the source ontologies and than to reorganize the required concepts under that ontology. In the integration case a number of integration operations are applied to the ontologies in order to obtain a shared model. Pinto (2000) distinguishes basic integration operations such as *create-class*, *change-instance-name*, *change-axiom-definition* or *remove-subontology* and non-basic integration operations such as *extension*, *mapping*, *concatenation* or *restructuring*. The latter are composed of a set of basic integration operations and are defined on a higher level of abstraction.

Argument provision The *argument provision* activity is part of the DILIGENT methodology. It is performed within the DILIGENT argumentation framework, which is described in Section 4.4. The argumentation framework supports the ontology engineer exchanging arguments about the development and evolution of ontologies. The argumentation framework defines a process for exchanging arguments, identifies major roles in that process and specifies a set of argument types which facilitate the agreement process.

Evaluation of shared ontology 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 (Uschold et al., 1998a; Gómez-Pérez, 2001; Guarino & Welty, 2002). Assessing the usability of an ontology in a target application context, the second category, is addressed for example in OntoMetric which is a framework for selecting appropriate ontologies (Lozano-Tello & Gómez-Pérez, 2004). 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 Documentation is a textual description of the ontology and its creation process. Ideally documentation is performed in parallel to the aforementioned activities in order to ensure an efficient process quality control and improve the reusability of the ontology.

Evaluation and *Documentation* are ontology support activities which should be performed in all stages of the process. As the methods are the same for all stages we do not mention these activities in the following detailed descriptions.

4.3.2. Local Adaptation

The initial ontology is released and distributed to the users and they start to adapt it locally for their own purposes. Note, that each user has a copy of the shared ontology. Changes are not introduced directly in the shared ontology, but in the local copy of the shared ontology, *i.e.*the *local ontology*. We call local ontologies from other users *remote local ontologies*. The users utilizing a new version of the shared ontology first get familiar with the shared ontology. In the next step they interact with the ontology in parallel in a threefold manner, depending on the concrete application setting. For example some users use the ontology only for retrieving information either locally or from other participants. Others also actively instantiate the ontology with their own knowledge. Finally, the use and the analysis of the shared ontology can result in the definition of new local requirements for the shared ontology and the change of the local ontology.

4.3.2.1. Roles

The local adaptation stage requires only the user role. The users' primary tasks are usually not related to ontology engineering but to their daily work. Therefore, users may not have an ontology engineering background or even experience. They may be experts for their specific tasks, but are not required to be domain experts for the entire domain of the shared ontology. In our case study they used the ontology to retrieve, *e.g.*, documents related to certain topics modeled in the ontology. They also search for more structured data, such as the projects an employee is involved in.

4.3.2.2. Input

The locally available information and the shared ontology are inputs for this stage. This information is used to populate the ontology and to change the local ontology. Local information can be existing databases, ontologies or folder structures and documents. The users may consult external information, such as the ontologies of other users, ontologies found in ontology repositories, and the literature.

4.3.2.3. Output

The output of the local adaptation stage is a locally changed ontology which better reflects the user needs. Each change is supported by arguments explaining the assumptions underlying the requested changes. The changes are not directly propagated to the shared ontology, but first undergo a detailed analysis in the *central analysis stage* by the board.

4.3.2.4. Decisions

The users decide upon the changes to their local version of the shared ontology. They decide if new concepts are needed and how to introduce them, if concept should be deleted or modified.[3]

4.3.2.5. Activities

In order to change the local ontologies the users follow different paths. Ontology users may decide to reuse existing ontologies originating from other users involved in the process or to conceptualize the desired changes themselves. As other users, in particular the board, should be able to understand these changes, argument provision becomes crucial. Figure 4.4 visualizes the dependencies between the single activities.

To obtained the desired output the user performs different activities namely *local analysis of shared ontology*, *local specification of new requirements* , *ontology use*[4], *local customization of local ontology*, *local integration of reused ontologies to the local ontology*, *local modification of the local ontology*, *argument provision*, *evaluation of new local ontology* according to the sequence depicted in Figure 4.4. The activities are repeated until a new shared ontology is available.

Local analysis of shared ontology In this activity the users get familiar with the shared ontology. Although an ontology is an explicit specification of a domain, not all assumptions underlying the design are formalized. Therefore, it is necessary in a first step to understand these assumptions. The users learn where the different concepts are located in the ontology and how they are interrelated with other concepts. It is not required that the user understands the entire ontology immediately. The analysis can be performed gradually. For our use case the methodology suggests two specific tasks.

[3] It should be stressed that when we talk about changing the ontology or introducing new concepts this applies to concepts, relations and axioms.

[4] Not visualized in the activity diagram, as ontology use is independent of the process state.

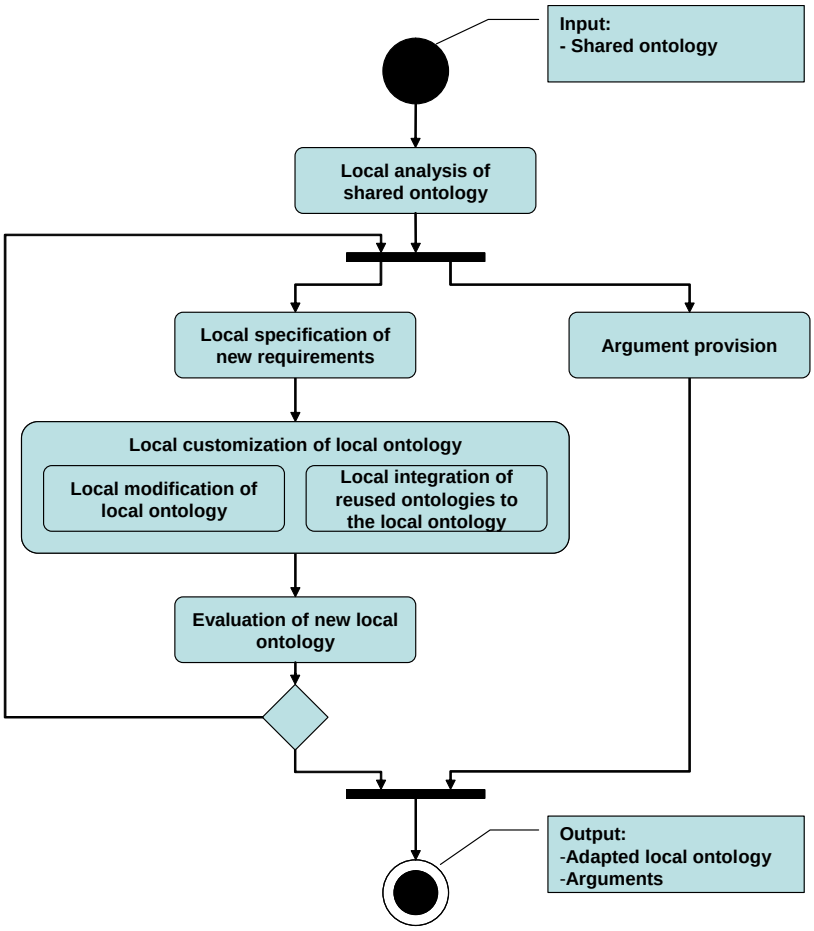


Figure 4.4.: Local Adaptation

User task:	*Understand shared ontology*
In order to understand the shared ontology the users first look at the different modules (implicitly) defined in the ontology. They can then look up the definitions of the concepts and read the available documentation. The arguments underlying the conceptualization may also be helpful. As a guideline the users 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 instances defined in the shared ontology may serve as good examples for the meaning of the instantiated concepts.	

User task:	*Identify commonalities between own and shared conceptualization*
In order to find commonalities between the shared ontology and local conceptualizations, users look for concepts and relations in the shared ontology which have the same names or synonyms of concepts in their local conceptualization. If they reuse their folder hierarchy in order to extend the shared ontology they compare the labels of folders with the labels of concepts and relations defined in the shared ontology. They understand the differences between the conceptualization in the shared ontology and their own.	

Ontology use As described in Section 2.2.1 an ontology can be used as a controlled vocabulary, for browsing support, for search support, for sense disambiguation, for consistency checking, for interpretability support and many more (McGuinness, 2003). Depending on the restrictions imposed by privacy laws, ontology usage can be monitored. Frequency of use of certain entities in the ontology can provide the necessary data to the board to enhance the quality of the shared ontology.

User task:	*Organize local knowledge according to the conceptualization*
Local information is one source of available knowledge. In the case studies described in this thesis the local information consisted of documents, contacts, emails, and Web bookmarks. Users populated the ontology organizing their local knowledge according to the shared ontology. All participants contribute to the collective knowledge available in the network. Users also integrate knowledge they retrieve from third parties into their own knowledge base. Relations between own and remote conceptualizations beyond the already shared ontology may be detected during the integration of external knowledge. If the users formalize knowledge which cannot be represented in the shared ontology this leads to the specification of new local requirements.	

Specification of new requirements The local usage of the shared ontology leads to the specification of new requirements, if it does not completely represent the knowledge required by the users.

The task(s) the ontology is involved in play(s) an important role in identifying the requirements. The possibilities are wide-ranging. As already mentioned in the build stage trained ontology engineers can use an ORSD to capture the requirements in this stage. Other methods to identify requirements have also been mentioned in conjunction with traditional ontology building efforts. Requirements can be derived from competency questions as it was suggested in (Grüninger & Fox, 1995). Existing ontologies may be the driver for new requirements, too.

The user may submit the requirements to a representative, if he does not want to customize the ontology on his own.

User task:	*Identify missing conceptualizations*
Mismatches between the concepts defined in the shared ontology and the local conceptualizations are starting points to identify missing conceptualizations in the shared ontology and specify local requirements. If available local conceptualizations are for example folder hierarchies folder names which cannot be mapped on concepts defined in the shared ontology may indicate missing conceptualizations.	

Local customization of local ontology In Section 4.3.1 we have introduced two ways of building an ontology: from scratch or by reuse or a combination of both. In the local adaptation stage the users also have these two possibilities to change the local ontology. As visualized in Figure 4.4 the local customization activity has the two sub-activities *Local modification of local ontology* and *Local integration of reused ontologies to the local ontology*.

Local modification of local ontology The local modification of the shared ontology is one option to adapt it to the local requirements. We distinguish two cases: (1) the shared ontology covers more domain knowledge than the user requires, and (2) the shared ontology covers less domain knowledge than the user requires. Without loosing generality this separation covers also the intermediated cases, if some parts of the ontology are too detailed while other are too coarse.

In the first case the user removes the conceptualizations from the shared ontology which he does not need.[5] This pruning activity has implications on later process stages, because the board draws conclusions from the user activities. If the board considers the removal of a concept from the shared ontology, it distinguishes between the removal of top level concepts and leaf concepts deep in the hierarchy. High level concepts may be too general to be of concrete use for a specific user, but facilitate interoperability, thus there removal may be counter productive. In contrast to this case, very specific concepts may be too specific for being part of the shared ontology. The removal of concepts from the shared ontology requires the board to make well-balanced and well-founded decisions.

If user requirements are not met by the shared ontology he modifies the shared ontology locally. Modifications range from small changes, *e.g.*, adding a new concept or relation to a complete restructuring of the local ontology. The local requirements can also entail that the user integrates a new module into the shared ontology. New local conceptualizations point the board to missing conceptualizations in the shared ontology.

As a result of this activity the local ontology fulfills the local requirements.

Local integration of reused ontologies to the local ontology The second possibility to meet new local requirements of the users is to reuse external ontologies. This activity is

[5] For this case Swartout et al. (1996) propose an independent methodology, guiding the user in the ontology pruning process.

thus related to the activities defined in ontology building by *reuse* (*cf.* Section 4.3.1.5) .

Depending on the application the user may have direct access to other party ontologies. User may reuse local adaptations to remote local ontologies. Reuse in this scenario is facilitated, because users reuse extensions to a shared ontology. The integration of entire or parts of remote local ontologies is preceded by an examination of the candidate ontologies. The users decide which integration operations, such as change-concept-name, remove-concept or add-relation are executed on the copies of the remote local ontologies before their local inclusion.

Reusing existing ontologies is a feasible alternative for both technically versed ontology engineers, who follow established reuse methodologies to fulfill this task, and less experienced ontology users. As the shared ontology offers a common structure the user can focus the examination of remote ontologies on the parts which are relevant for him instead of analyzing the entire remote ontology. Although remote ontologies may not exactly meet user requirements they might prefer to reuse them, because it increases interoperability.

If the user has found reusable external ontologies he merges or integrates them with his local ontology. Depending on the reuse level the remote ontologies may be subject of more or less radical customization. The addition of generalizations or refinements may be needed in order to integrate the reused ontologies. The effort locally invested to reuse ontologies from external parties are compensated by the fact the board can more easily detect those parts of the ontology which are likely to be shared.
Additionally to the reuse of remote ontologies users can also provide mappings to remote ontologies. The users benefits from mappings, because they can access remote users data. The board can recognize parts of the ontology which are again likely to be shared.
The result of this activity is a locally adapted shared ontology with adaptations based on remote user ontologies.

DILIGENT defines two user tasks for this activity.

User task:	*Retrieve extensions to the shared ontology from other users*
Users retrieve the changes to the shared ontology form other users. They look for extensions to top-level concepts defined in the shared ontology. They include *all*[6] sub concepts of shared top-level concepts defined in the remote local ontology, if they choose to reuse it for their own purposes.	

User task:	*Map equivalent conceptualizations*
Users reuse locally available conceptualizations. They either reuse these conceptualization in order to extend the local ontology or they provide mappings between the shared ontology and the local conceptualizations.[7]	

[6] This is a simplification of the general reuse case, in order to reduce the conceptual burden on the users in the SWAP use case.

Argument provision The users externalize the reasons for their modeling decisions using arguments following the procedures proposed in the argumentation framework (*cf.* Section 4.4).

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 submitted change request to the shared ontology. The provision of arguments has the further advantage that users intending to reuse the conceptualization – as aforementioned in the previous, reuse-oriented activities – better understand the assumptions underlying the remote local ontologies.

Evaluation of new local ontology The evaluation activity is described in Section 4.3.1. The user evaluates his local ontology w.r.t. his local requirements. He does not evaluate the entire ontology, but only the parts relevant to him. The local ontology should evolve with changing requirements.

Documentation As far as possible the user should document the changes introduced into the shared ontology. Documentation includes metadata provision, such as the change time and its author. Brief descriptions of the added conceptualizations facilitate the analysis task of the board.

No user tasks are defined for the last three activities.

4.3.3. Central Analysis

The board meets at regular time intervals or if a certain threshold of change requests has been reached. The meeting frequency depends on the ontological capabilities of the users and the volatility of the domain. The user changes cannot be directly integrated to the shared ontology, because changes from different users may be in conflict with each other. If the shared ontology should respond to new requirements quickly the board meets frequently. The total number of changes introduces by the users is an indicator for the urgency of a board meeting. Frequent board meetings which gradually introduce changes reduce the overall effort, because all users profit from the up-to-date version of the shared ontology and need not to rely on possibly contradictory conceptualizations found in remote local ontologies. If the modeling capabilities of the users are high, more remote local ontologies can be reused, hence up-to-date conceptualizations are available without being immediately integrated into the shared ontology.

[7] Section 4.5.2 presents a tool supporting the manual creation of mappings between ontology entities and automatic support to find mapping candidates.

In order to maintain an up-to-date version of the ontology for volatile domains, with frequently emerging new requirements, the board should meet more often than in more stable domains.

In each meeting the board gathers the user local ontologies. The board analyzes incoming requests and observations of changes. The central analysis stage is carried out in conjunction with the subsequent central revision stage. The separation indicates the differences between the identification of new requirements in the analysis stage and their implementation in the revision stage. In the revision stage the user representatives are not required.

4.3.3.1. Roles

In the central analysis stage we can distinguish three roles played by board members: (i) The domain experts choose among the requested changes the ones which are relevant for the shared ontology. (ii) Representatives of the users justify different requirements 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.[8] Board membership should be open to all users of the ontology.

4.3.3.2. Decisions

The board decides which changes to introduce into the new shared ontology at the conceptual level. It may found this decision on the usage patterns observed for the ontology, such as the number of users who introduced a change in proportion to all users who made changes, the number of queries including certain concepts or the number of concepts adapted by the users from previous rounds. General organizational requirements which are not observed on the local level may also lead to changes in the shared ontology.

4.3.3.3. Input

The board starts the central analysis with the previous version of the shared ontology and takes as input the local ontologies and the user change requests. To be able to understand the change requests, users should provide arguments for each request. The arguments underlying the proposed changes constitute important input for the board to achieve a well balanced decision about which changes to adopt.

[8] In the central revision stage.

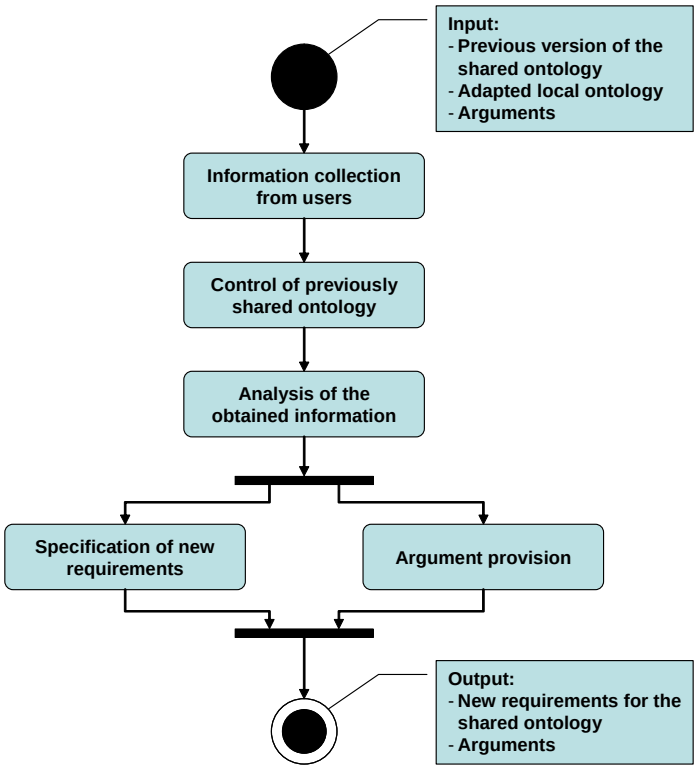


Figure 4.5.: Central Analysis

4.3.3.4. Output

The result of the central analysis stage is a list of the new requirements and relevant change requests for the shared ontology. In this stage the board considers changes only on the conceptual level. The clear separation of the identification of change requests on the conceptual level from their implementation is advantageous, because the local ontologies probably contain many different formal change requests which are conceptually similar. The separation emphasizes the need to look for conceptual similarities first and then think about their implementation.

4.3.3.5. Activities

The board meets regularly in order to include emerging requirements into the shared ontology. To achieve the desired output the board takes different activities namely *Information collection from users*, *Analysis of obtained information*, *Control of shared ontology*, *Specification of new requirements* and *Argument provision* as visualized in Figure 4.5.

We now detail each one of the proposed activities:

Information collection from users The board collects the user local ontologies, the argumentation, change requests provided by other means, usage information and finally mapping information. They may also collect local conceptualization, such as folder structures, if the users are willing to share them with the board.

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

We have defined a number of controlling measures to support the board.

Adaptation rate The adaptation rate of an ontology entity calculates the proportion of users utilizing it in comparison to all users of the system. It is an indicator for the acceptance of an entity. $NoLocalInclusions$ represents the number of local ontologies which include the ontology entity $OntologyEntity$. $TotalNoUsers$ is the total number of users utilizing the shared ontology. The adaptation rate is defined according to Equation 4.1. If the users do not accept an entity of the core ontology it should probably be changed. A popular entity introduced in a local ontology of a user which has been reused by many others may be integrated into the shared ontology. The possible conclusions from comparatively high or low $AdaptationRate$s depend on the application scenario and include the deletion, introduction, reorganization or renaming of the respective ontology entity.

$$\text{AdaptationRate}_{OntologyEntity} := \frac{NoLocalInclusions_{OntologyEntity}}{TotalNoUsers} \tag{4.1}$$

Usage rate The usage rate refers to the different aspects of ontology use, *e.g.*, the number of instances created for a class, the number of queries containing a specific entity, the number of sub-concepts defined for a concept (Equation 4.2). $NoUsages$ refers to the number of users who have used an ontology entities at least once. The ratio between the number of users of a particular ontology entity and the total number of users ($TotalNoUsers$) is defined as the $UsageRate_{OntologyEntity}$ of an ontology entity. The implications of a high usage rate are domain specific. It suggests that the respective ontology entity is relevant. It may suggest that an ontology entity should be further elaborated.

$$\text{UsageRate}_{OntologyEntity} := \frac{NoUsages_{OntologyEntity}}{TotalNoUsers} \tag{4.2}$$

Change rate The change rate is defined as the average number of modification to the shared ontology by the users. A high change rate indicates, that the users are not

satisfied with the shared ontology as a whole and that they extensively reorganize it. In contrast a low change rate suggests that the modeling decisions of the board did capture the user requirements; in particular when the low changes rate is observed in conjunction with a high average usage rate.

The activity *Control of shared ontology* is not part of the tested methodology, but has been introduced as a result of the case studies. Previously the adaptation rate calculation was part of the *Analysis of obtained information* activity. The *control of shared ontology* activity separates the backward oriented analysis from the change oriented analysis. This is the objective of the next activity.

Analysis of obtained information The analysis of the obtained information should identify the parts of the shared ontology which should be modified. First the board identifies the different areas in which changes took place. The board should analyze changes of concepts, before changes of relations and these before changes of axioms.

The board may understand the change requests better if analyzed in the same order as they have been applied instead of only looking at the final user ontology. With an increasing number of participants this in-depth analysis may be unfeasible.

User task:	*Analyze the introduced changes*
The board analysis the retrieved ontologies, the requests from the users and other available information, *e.g.* folder structures. Indicators for changes relevant to the users are (i) overlapping changes and (ii) their frequency. Furthermore, the board analyzes (iii) the queries made to the ontology. This helps to find out which parts of the ontology were frequently used. Since actors instantiate the ontology locally, (iv) the number of instances for the different proposed changes may be used to determine the relevance of certain adaptations.	

Argument provision This activity is described with more detail in Section 4.4.

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. The new requirements should be captured in an updated version of the ORSD. In contrast to the specification of requirements in the initial build stage, the board can derive requirements from the locally adapted ontologies.

User task:	*Identify changes presumably relevant for a significant share of all actors*
Having analyzed the changes and having grouped them according to the different parts of the ontology they belong to, the board identifies the most relevant ones. Based on the provided arguments the board decides which changes should be introduced. Depending on the quality of the arguments the board itself may 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 is a reduced and structured list of changes that is applied to the shared ontology.	

4.3.4. Central Revision

Central analysis and central revision are joint stages. While in the analysis stage the board defines new requirements, they are formalized and implemented in the revision stage. In the end of the revision stage the new shared ontology is distributed to the users.

4.3.4.1. Roles

The ontology engineer judges the changes from an ontological perspective. Some changes may be relevant for the common ontology, but may not be adequately formalized by the users. The domain experts should judge and decide whether new ontology entities should be introduced into the common ontology.

4.3.4.2. Input

The input for the central revision stage is a list of new requirements which should be included into the shared ontology and the arguments underlying the new requirements.

4.3.4.3. Decisions

The decisions taken in the central revision stage are formal ones. All intended requirements decided during the analysis stage should be included into the common ontology. In the revision stage the ontology engineer decides how the requirements can be met formally. Evaluation of the decisions is performed by comparing the requirements with the final formal decisions.

4.3.4.4. Output

The outcome of the central revision stage is a new and documented version of the shared ontology. The introduced changes are supported by arguments.

4.3.4.5. Activities

The board members perform the activities *Customization of shared ontologies*, *Integration of reused local ontologies to the shared ontology*, *Modification of shared ontology*, *Argument provision*, *Argumentation aggregation*, *Evaluation of new shared ontology*, and *Distribution of new shared ontology* in the sequence depicted in Figure 4.6.

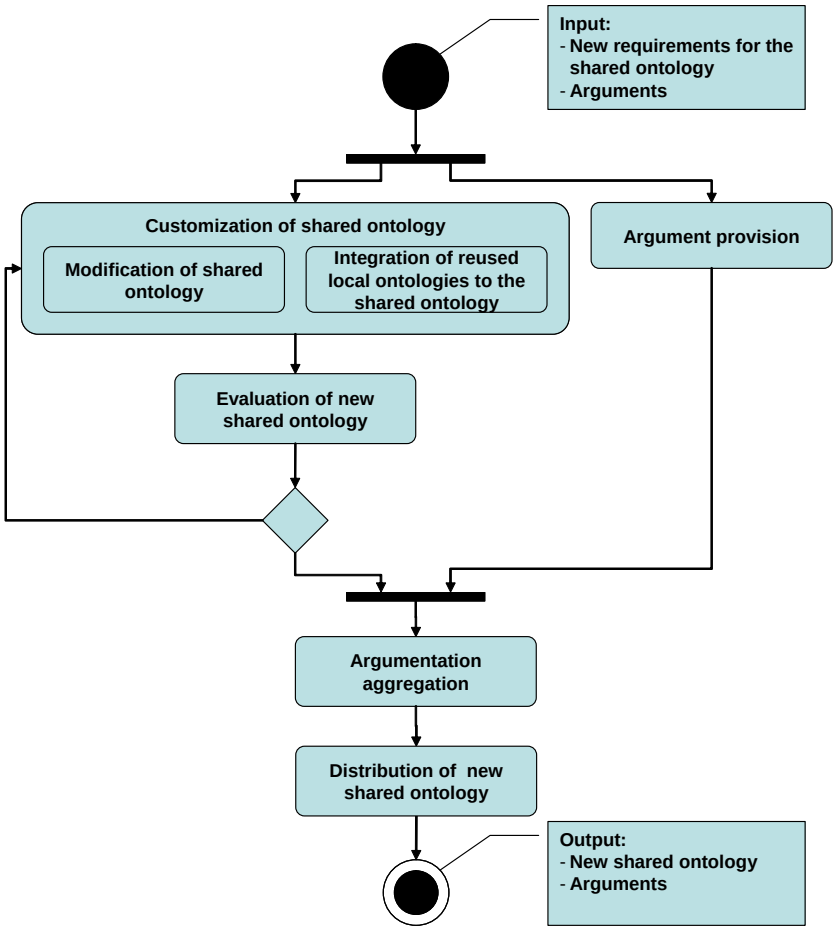


Figure 4.6.: Central Revision

Customization of shared ontology Changes to the shared ontology can be introduced through modification or by reusing user formalizations.

For our use case we defined only one action.

User task:	*Formalization of the decided changes*
In our use case the board should formalize the relevant changes identified in the previous stage. The labels for concepts and relations proposed by the users should be reused if adequate. The board introduces generalizations if required.	

Integration of reused local ontologies to the shared ontology If user local ontologies can be reused in order to meet the new requirements, this activity is similar to the *reuse* activity in the central build stage. The board integrates the customized local ontologies with the shared ontology. It may be necessary to include abstractions or refinements into the shared ontology in order to integrate the reused local ontologies adequately. For later activities it is helpful to trace the origins of reused ontology entities in that users updating their local ontology with new shared ontology know whether their requests have been addressed.

Modification of shared ontology If new requirements cannot be addressed by reusing local ontologies the board modifies the shared ontology. This activity is similar to the initial build stage. The board should take into account the general guidelines defined in the ORSD when new concepts are added to the shared ontology or concepts are modified or deleted.

Evaluation of new shared ontology The evaluation of the new shared ontology is carried out in the same manner as in the initial build stage (*cf.* Section 4.3.1.5).

Argumentation aggregation The board exchanges arguments in the decision process. The changes eventually included into the shared ontology are therefore supported by 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. In order to make the arguments better understandable to the users the board aggregates them.

Distribution of new shared ontology The board distributes the updated shared ontology to the users. Depending on the overall system architecture different methods, such as publication on a Web site, distribution via an advertiser in a P2P system, or distribution via email, may be applied.

4.3.5. Local Update

As a result of the central revision stage the users have access to the new version of the shared ontology. They decide which parts - if any - they use from the new shared ontology. Switching from the local ontology to an updated shared ontology implies effort for understanding the new parts and partly reorganizing the local knowledge base. The gains of updating are, first lower communication effort and, second up-to-date information. The incentives for the user to update are higher the more of his change requests to the shared ontology are included in the new shared one. Thus the user reviews how many of his own proposals are included in the new version and how they have been implemented. Eventually the user decides whether to integrate the new version with his local ontology.

4.3.5.1. Roles

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

4.3.5.2. Input

The new shared ontology and the arguments exchanged by the board are the input for this stage. The new shared ontology contains a list of new concepts.

4.3.5.3. Decisions

The users decide which changes they introduce locally. This depends on the differences between the own and the new shared ontology. The users do not need to update their complete ontology.

4.3.5.4. Output

The output of the local update stage is an updated local ontology which may include changes made to the shared ontology. It is not required that the users apply all changes proposed by the board.

4.3.5.5. 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* as illustrated in Figure 4.7.

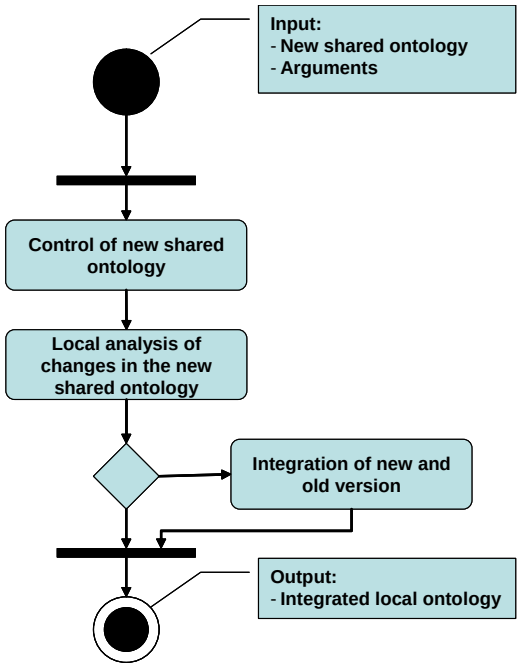


Figure 4.7.: Local Update

Control of new shared ontology Likewise the board the users control the implementation of their own proposals. The users control whether the board has implemented their requested changes in the new shared ontology. This allows the users to judge which of his proposal are interesting for the community. Furthermore, they learn how the board translates their proposals into conceptualizations in the shared ontology.

We have defined a number of measures to support the user controlling the new shared ontology.

Overlap measure The overlap measure is calculated for each user according to Equation 4.3 and is the ratio between the number of all user requests ($|TotalLocalRequests_{User}|$)[9] and the number of actually integrated requests ($|IntegratedChanges_{User}|$). The number of $IntegratedChanges$ is the sum of changes (*cf.* Equation 4.4) which have been directly integrated as proposed by the user ($DirectlyIntegratedChanges_{User}$), which have been conceptually integrated ($ConceptuallyIntegratedChanges_{User}$), and which have been integrated due to informal user requests ($IntegratedRequests_{User}$).

$$OverlapMeasure_{User} = \frac{|IntegratedChanges_{User}|}{|TotalLocalRequests_{User}|} \tag{4.3}$$

[9]We count each change to the shared ontology and each issue sent to the board as one request.

$$|IntegratedChanges_{User}| = |DirectlyIntegratedChanges_{User}| + |ConceptuallyIntegratedChanges_{User}| + |IntegratedRequests_{User}| \quad (4.4)$$

Direct overlap measure The direct overlap measure is the ratio between the number of user changes to the shared ontology and the ones integrated in the shared ontology as proposed by the user (*cf.* Equation 4.5).

$$DirectOverlapMeasure_{User} = \frac{|DirectlyIntegratedChanges_{User}|}{|TotalLocalRequests_{User}|} \quad (4.5)$$

Conceptual overlap measure The conceptual overlap measure is the ratio between the number of all user changes to the shared ontology and the ones conceptually integrated in the shared ontology. If the shared ontology has been modified in order to represent the change made by the user, but the modification is different from a formalization point of view, the change has been integrated conceptually. For example the introduction of intermediate concepts in the shared ontology, changes the user formalization but not the intended meaning (*cf.* Equation 4.6).

$$ConceptualOverlapMeasure_{User} = \frac{|ConceptuallyIntegratedChanges_{User}|}{|TotalLocalRequests_{User}|} \quad (4.6)$$

This activity was not validated in the case study, and it remains future work to define more controlling measures. The requests of the user, however, motivated the introduction of the activity into the general methodology.

Local analysis of changes in the new shared ontology The users locally change to the new shared ontology only if their benefits outweigh the effort of updating. The analysis of the introduced changes informs them whether the changes affect them or not. The users should first analyze the parts of the ontology which are relevant to them. The users should start the examination with the additional concepts and their documentation, they should continue with the examination of additional properties and finally examine the new axioms.

Integration of new and old version The result of the analysis is the decision whether or not to integrate completely or partially the new shared ontology with the existing local ontology. The new shared ontology may contain refinements of the existing model. In this case the user should consider adaptin his instantiations with respect to the refinements. The outcome of the controlling activity allows the user to judge the restructuring effort in order to stay in line with the new shared ontology.

The new version can also model 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.

For our use case we have defined three tasks in order to overcome some of the technical challenges of our prototypical implementation.

User task:	*Tagging of the old local ontology*
The users store the old version of the ontology before they include the new shared version. They should preserve the date of the update and the version.	

User task:	*Local inclusion of the updated version*
The users include the new version of the shared ontology locally. They should examine the new version and look for extensions.	

User task:	*Alignment of old and new versions*
The users align their local adaptations with the new concepts defined in the shared ontology if required.	

After the *local update* took place the iteration continues with the *local adaptation* stage. In the next *central analysis* stage the board reviews which changes have been accepted by the users.

4.4. The DILIGENT Argumentation Framework

In DILIGENT the externalization of the assumptions underlying the conceptualization of the shared ontology and the changed local ontologies is of particular interest, because the users are locally dispersed, so that the board can neither directly explain its assumptions to the users nor can the users directly explain theirs to the board or other users. Users are allowed to enter or leave the ontology development process at any moment, therefor some users are not aware of the development history, while the history aware users might leave the process. An evolution of the shared ontology, though, without considering historical design decisions might have unpredictable consequences.

In DILIGENT the construction of the shared ontology is a community process, in which the ontology builders exchange arguments in order to reach agreement for a conceptualization. In order to address the above mentioned issues DILIGENT foresees to capture the exchanged arguments in a semi-formal way. This has the following advantages for the ontology engineering process:

- The availability of arguments in favor and against certain design decisions allows the ontology users to extend and change the shared ontology adequately, because design decisions can be communicated to other users.
- Tracing the arguments for design decisions prevents from the repetition of discussions related to the same issue.
- New users may enter the development process and can recapitulate the ontology creation history and the decision process.
- A structured provision of arguments guides the design process.
- A semi-formal representation of the argumentation allows for the detection of inconsistencies in the user argumentation.
- An ontology with externalized assumptions is more reusable.

The semi-formal provision of arguments is embedded in the DILIGENT argumentation framework consisting of a process description for the provision of arguments and an argumentation model.

4.4.1. Arguments in the DILIGENT Process

A central issue in the DILIGENT process is keeping track of threads of exchanged arguments. As described in Section 4.3 arguments play an essential part in all stages:

- The board exchanges arguments while **build**ing the initial shared ontology in order to reach consensus towards "a shared specification of a conceptualization" (Gruber, 1995);
- 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 be **revised**.

4.4.2. The Argumentation Process

The participants in the argumentation process take different roles in the ontology engineering discussion. They can profit from information generated in preceding activities as input to the discussion in order to obtain the desired output.

4.4.2.1. 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.

4.4.2.2. Input

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

4.4.2.3. Output

The objective of an argumentation process for DILIGENT 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.

4.4.2.4. 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 arguments, though. New issues can emerge, while for others solutions have already been defined.

Choose moderator The participants in the ontology engineering discussion choose a moderator. The basic rules for moderation also apply for ontology engineering discussions: the moderator does not contribute to the discussion, but structures it; he does not take part in decision, but organizes the decision process.
Any participants may take the role of the moderator and the moderator role may move from one participant to the next. The moderator should, however, have the necessary respect from all participants so that he can enforce the rules.

Choose decision procedure The participants agree on a decision procedure. First, they agree on a voting mechanism to select among a number of choices (*cf. e.g.*, (Arrow, 1963; Wikipedia, 2006b)). DILIGENT proposes the following mechanisms:

Majority voting All participants have the same voting weight and one vote. An issue is accepted if more than, *e.g.*, half of the participants accept it.

Preferential voting All participants have the same voting weight and have more than one vote. They express their preferences with the number of votes they allocate to the different solutions.

Dictator model One participant decides if an issue is accepted or not.

Second, they 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. The selected voting mechanism depends on the application scenario and the user group.

Specify issues The issues define the conceptual requirements for the ontology. The participants begin with the identification of relevant issues. For example, the requirements defined in the ORSD and the competency questions may be used to identify initial issues. During the discussion elaborations of the initial set of issues are introduced. In particular should any participant be allowed to add new issues or elaborate on existing issues. The decision, whether an issue is relevant is not discussed during the issue generation phase. At later stages issues are grouped according to their priority for the application and treated accordingly. In particular in online 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. The issues should be grouped according to the subdomains they belong to. Issues which are specifications or elaborations of already existing ones, should be classified as elaborations.

Provide arguments and ideas The participants discuss the issues, and bring forward their arguments in favor or against an issue. The suggested 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 first agree whether an issue is relevant for the ontology under discussion, before they start to model it formally. If a formalization depends on more than one issue, the participants first agree on all related issues before their formalization. Arguments should only be brought forward once instead of repeating an argument in other words. The participants should, however, express their agreement or disagreement with arguments in order to strengthen or weaken them.

In case of conflict the participants should first determine the type of conflict. The participants should express the exact meaning of a concept and define its properties to determine whether they talk about the same things with the same terminology. Awareness of the conflict type facilitates the discussion and the conflict resolution. Shaw & Gaines (1989) describes different types and Aschoff et al. (2004) adapts them to ontologies as displayed in Table 4.1. The classification distinguishes concepts and terminology and people can talk about the same or different concepts while using the same or different terminology. If two

persons talk about the same concept with the same terminology they agree and reach *consensus*. In another case they use different terminology for different concepts and produce a *contrast*.

Concepts \ Terminology	same	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 4.1.: Types of Conflict (Shaw and Gaines, 1989)

Decide on issues and ideas The participants may agree or disagree on an issue or postpone its discussion. Only ideas which are agreed are part of the shared ontology.

4.4.3. Building an *Argumentation Ontology*

As motivated above arguments in DILIGENT are captured in a semi-formal way. The arguments are represented semi-formally, opposed to formally, because the arguments themselves are not formalized. Users provide their arguments in natural language, but categorize them depending on their intention. In that the argumentation is not completely unstructured. The *Argumentation Ontology* models the main concepts relevant in an OE discussion. In the following *domain ontology* refers to the ontology under discussion, thus the ontology which is enhanced with arguments modeled according to the *Argumentation Ontology*.

The following Sections concentrate on the development of the *Argumentation Ontology* and describe its *specification*, *conceptualization* and *formalization*.

4.4.3.1. Specification of the *Argumentation Ontology*

We have derived the requirements for the DILIGENT *Argumentation Ontology* from literature research, the scenario setting, and design guidelines for ontologies (Gómez-Pérez et al., 2003).

1. **Use established terminology** Research in argumentation and its visualization is a mature field (*cf. e.g.*, (Conklin et al., 2001)). Acceptance of the ontology depends on the use of an established terminology.

2. **Focus on relevant arguments** The available types of arguments should be restricted such that the OE discussions are focused and efficient. This view is supported by Buckingham Shum et al. (2003) who have developed an ontology for a different domain, but for a similar purpose and found that usability benefits from a smaller ontology.

3. **Ontology focus** Following the results of Potts & Bruns (1988) general purpose argumentation models, such as Issue Based Information Systems (IBIS), should be extended with domain specific knowledge for their application in specific use cases. Therefore, the developed *Argumentation Ontology* should be customized for ontology design. In particular the detection of contradictory information in the domain ontology is important. Contradictory information leads to inconsistencies in the formal model of the domain ontology. Table 4.2 enumerates three types of inconsistencies, which may be detected by representing arguments in a semi-formal way.

Inconsistency	Description
Idea inconsistency	In response to different issues a user proposes changes to the ontology. The formalization of the proposed changes renders the ontology inconsistent.
Argumentation inconsistency	Arguer argues first in favor and then against an issue.
Position inconsistency	If a user agrees with contradictory issues, he introduces a position inconsistency.

Table 4.2.: List of Argumentation Inconsistencies

4. **Adaptivity** The *Argumentation Ontology* should allow for capturing the structure of argumentation. Hence, the design should 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 argumentation process** The *Argumentation Ontology* should support the full argumentation cycle. This includes issue raising, conflict mediation, bargaining, clarification and agreement. Participants also aggregate lines of reasoning in order to make their argumentation more concise. 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 implementation. The model should support argumentation on both conceptual and formal level.

7. **Minimal ontological commitment** Minimal ontological commitment ensures the extensibility of an ontology for future uses (Gruber, 1993b). An ontology should

model only one domain at a time and the ontology engineers should avoid the integration of two many different issues. If an ontology integrates knowledge from different domains, this is achieved by modularizing the single domains. That means for the *Argumentation Ontology* that it should focus on the argumentation related ontology entities.

8. **Minimal encoding bias** The *Argumentation Ontology* should be independent of the formalism used to model the final ontology. Each formalism allows different sets of modeling decisions and all can be subject to discussion.

9. **Process awareness** The *Argumentation Ontology* is embedded into the DILIGENT process presented in Section 4.2. Essential properties of this process are its collaborative aspects, its distributiveness and the asynchronous way participants can provide arguments.

10. **Argumentation formalization** If participants decide to bring forward their arguments in a formal way these formalized arguments should be representable in the *Argumentation Ontology*. This will enhance the ability to detect inconsistencies in the lines of reasoning.

4.4.3.2. Conceptualizing the *Argumentation Ontology*

For each of the requirements we provide the necessary background information which eventually enables the conceptualization of the *Argumentation Ontology*.

Use established terminology Requirement 1 states that the terminology used in our ontology corresponds to the established ones. Although the Toulmin model (Toulmin et al., 1984) of argumentation was the first one presented in the literature, today Issue-Based Information System (IBIS) is the model of choice to represent argumentation processes (*cf.* Section 2.3). The *Argumentation Ontology* integrates the concepts *Issue*, *Idea*, *Argument*, *Challenge* and *Justification* from the IBIS model. In the *Argumentation Ontology* an issue represents a conceptual requirement for the ontology under discussion. Instead, an idea represents an implementation proposal to meet a certain requirement. An argument expresses an opinion about an issue or idea. Challenges and justifications are arguments against or in favor of an issue or an idea.

Focus on relevant arguments As described in Section 4.4.4 we have analyzed ontology engineering discussions in order to fulfill requirement 2. We identified the argument types *Elaboration*, *Example*, *Justification and Evaluation*, *Counter Example* and *Alternative* as the ones focussing the participants the most.

Ontology focus Ontology building starts with the specification of requirements. Requirements are then conceptualized and finally formalized in the target representation language. The *Argumentation Ontology* allows to discuss the design of the ontology on these three levels separately. This enables the participants to agree on the relevant issues independently of their conceptualization or formalization and vice versa. This reflects also the user roles in the DILIGENT process. Domain experts raise issues while ontology engineers propose ideas to solve the issues.

In order to detect inconsistencies in the user argumentation the *Argumentation Ontology* is implemented in OWL DL.

Adaptivity According to requirement 4 the *Argumentation Ontology* contains a general argumentation concept in order to capture the line of reasoning of a particular actor in the process. Depending on the granularity of the actor arguments he can choose among different sub-classes of the argumentation concept. Each sub-class can offer exactly the level of detail the actor wishes to provide at a certain time.

Support entire argumentation In order to organize and focus the participants in the discussion we elaborate on the concept decision. We introduce specializations of that concept to represent the decisions to accept, to reject, to postpone or to discuss an issue, an idea or an ontology entity (*cf.* Req. 5). The participants can decide on each level independently. However, an issue is only finally resolved if the participants either agree upon its conceptualization and formalization or reject it. If the partakers in the discussion realize that an issue cannot be resolved adequately they will postpone it, to ensure the progress of the building effort.

The stakeholder can agree or disagree with the arguments of other users. The agreement of many with an argument increases the weight of that argument and will lead to a corresponding judgement w.r.t to the issue, idea or ontology entity.

Conceptual as well as formalization level As stated above the design of the ontology starts with the requirements specification and ends with its formalization. For each requirement there can be a number of equally acceptable proposals for their conceptualization, while a conceptual model may be implemented in different ways. A separation of the three levels allows to discuss and agree at each level separately.

Minimal ontological commitment The *Argumentation Ontology* models primarily concepts which are relevant for the ontology engineering discussion. Other metadata oriented information should be included reusing other ontologies built for that particular purpose. For instance the *Argumentation Ontology* does not fine-granularly model the actor or ontologyEntity concept.

Minimal encoding bias The *Argumentation Ontology* does not include a formal representation of ontological constructs which are specific for a certain representation language.

Process awareness DILIGENT supports the user involvement in the ontology evolution, because they may change the ontology locally. It is not guaranteed that only one consistent ontology exists in the application, but local ontologies may be mutually inconsistent. In order to distinguish between ontologies from different user the change requests are associated with the providing actor. An issue is also associated with a status to separate the agreed model from the one under discussion. Additionally users may introduce new issues as elaborations of existing issues to state explicitly that they intent to extend the existing model instead of building a new one.

Argumentation formalization The the provision of arguments in a completely formal way is not specifically supported.

4.4.3.3. Formalizing the *Argumentation Ontology*

The DILIGENT *Argumentation Ontology* is visualized in Figure 4.8. 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, for instance as a class, an instance, a relation or an axiom. They relate to concrete ontology change operations.[10] Ideas are related to issues in the sense that they *respond to Issue*s. Ideas refer to how issues should actually be implemented in the ontology. In this way discussions can take place in both the conceptual level and the formalization level (Req. 6). Arguments are *arguments 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, which *elaborate on* an existing issue.

[10]For example (Stojanovic et al., 2002) presents a formal model for ontology change operations.

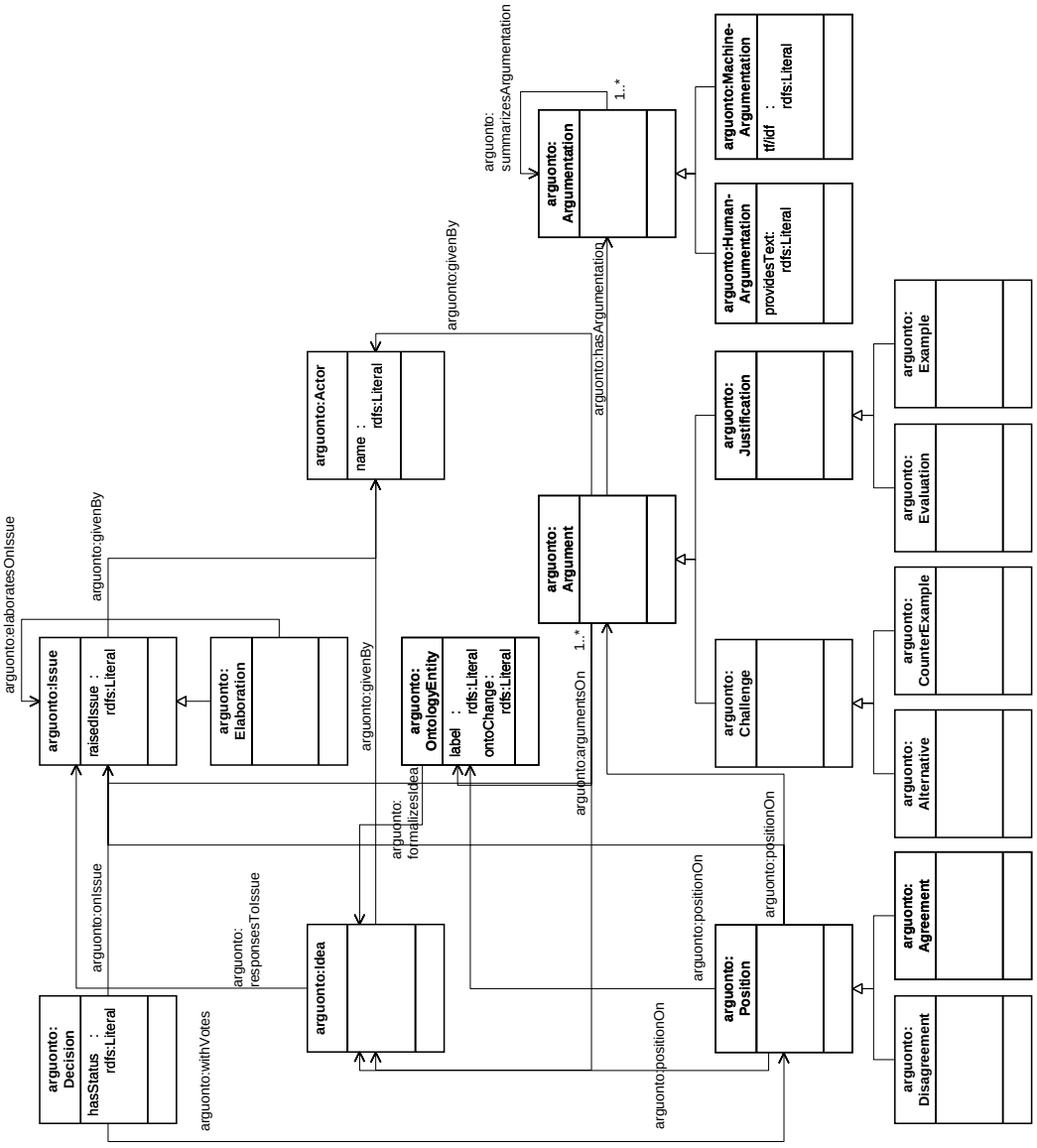


Figure 4.8.: DILIGENT Argumentation Ontology

Concepts	Relations →Domains	Description	Req.
Actor	hasName	Actors are participants in the process and can be humans as well as machines	4, 7
Argument	argumentsOn →{Issue; Idea; OntologyEntity}	Arguments can be provided on Issues, Ideas and OntologyEntities.	1, 5
	givenBy →Actor	Arguments can be issued by different actors	
	hasArgumentation →Argumentation	Arguments follow a line of reasoning	
└ Challenge			1, 2
└ Alternative		An alternative offers a different way to resolve an issue	1, 2, 3
└ Counter-example		A counter example provides evidence that s.th. cannot be modeled in the proposed way	1, 2, 3
└ Justification			1, 2
└ Evaluation		An evaluation justifies a modeling proposal by means of accepted evaluation criteria	1, 2, 3
└ Example		An example is a concrete instantiation for a modeling proposal	1, 2, 3
Argumentation	summarizes-Argumentation →Argumentation	An argumentation can either stand alone or summarize a line of reasoning	5
└ HumanArgu-mentation	providesText →Literal	An argumentation can play different roles, such as an alternative or an example. Therefore, the argumentation is separated from the argument.	4
└ MachineArgu-mentation	*depending on used Algorithm*	An algorithm based on TF/IDF and frequency could have these two properties.	4
Issue	givenBy →Actor	The issue is raised by a participant in the process	1, 5, 6
	raisedIssue	A high-level description of the requirement solvable by the issue	
└ Elaboration	elaboratesOn-Issue →Issue	A more detailed description of the issue	1, 2, 9
Idea	givenBy →Actor	An idea is proposed by an actor	1, 5, 6
	responseToIssue →Issue	An idea resolves a certain issue on the conceptual level	
OntologyEntity	formalizesIdea →Idea	Depending on the formalism, appropriate ontology entities formalize the corresponding idea	3, 6, 7, 8

Concepts	Relations →Domains	Description	Req.
Position	position-on →{Issue; Idea; OntologyEntity; Argument}	Participants can express their agreement or disagreement with Issues, Ideas, OntoloyEntities and Arguments	1, 5
└ Agreement		Sub-concept of Position to express agreement	1
└ Disagreement		Sub-concept of Position to express disagreement	1
Decision	withVotes →Position	A decision in favor or against an Issue, an Idea or an OntologyEntity. User positions support the decision with votes.	1, 5
	hasStatus →Literal	A decision expresses either an *Acceptance*, a *Rejection*, that the discussion was *Postponed* or that it is *UnderDiscussion*	9
	onIssue →Issue	This relation points to the respective Issues.	

Table 4.3.: DILIGENT *Argumentation Ontology*

The concepts an ontology represents 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 exchanged, 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 DILIGENT proposes examples and evaluation&justification as particularly useful argument types. Two classes in challenges are also particularly used in OE discussions: counter examples and alternative&contrast (Req. 3).

Those involved in discussions can state positions. They clarify the *position on* one issue, one idea, or an argument under discussion. Either they declare their agreement or disagreement. 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 to focus on the relevant arguments (Req 2).

Arguments are *given by* actors (Req 9). Humans or Machines can take the role of an actor. Different actors provide different argumentations (Req. 4). In what regards Argumentation, humans (ArgumentationHuman) tend to argue by providing strings of text stating (provides text) their reasons while machines tend to use other kinds of argumentation, *e.g.*, measures like Frequency and TFIDF (Maedche & Staab, 2001). For each algorithm, new

subclasses of ArgumentationMachine need to be introduced to model the different kinds of measures.

4.4.4. Argument Selection for the *Argumentation Ontology*

Although IBIS is a general model for argumentation it was conceived and further developed mainly in the context of requirements engineering and software design. Ontology design differs from software design, because an ontology is a shared conceptualization of a domain requiring agreement among its users; reusability and evolution of the ontology are more important in ontology design than software design; an ontology often represents content and domain knowledge whereas software encodes application knowledge (Devedžić, 2002).

The differences are even more obvious w.r.t. the DILIGENT process. The process supports the users to change the shared model and requires that they support the changes with arguments. In contrast software is changed by its users only in rare cases. Meanwhile in DILIGENT the users change the shared model locally so different versions of the same model coexist within the application. In order to enable collaboration even under these conditions, the users should explain their changes, thereby convincing other users and the board, that they need them and that they are useful.

Due to these differences, an extension to the IBIS model is necessary to better support ontology design discussions. In particular we focus on the exchanged argument types. In the following we describe the methodology and present the results of our experiments to identify the argument types which focus the discussion. The *Argumentation Ontology* extends the generic IBIS model with the argumentation types identified in these experiments, *i.e.*, the concepts Justification and Challenge. The experiment itself is described in Section 5.4.2.

4.4.4.1. Methodology

This section describes the methodology which was followed in order to investigate whether some argumentation structures dominate the progress in the OE task and should therefore be accounted for explicitly.

Three experiments build the basis for the argument selection. Section 5.4.2 elaborates on them with more detail, while this section emphasizes the results of the experiments. The first experiment consisted of an analysis of a well documented historic example of ontology construction. The second experiment supervised a small group of domain experts building an ontology without any guidance. The last experiment asked a small group of domain experts to use mainly specific arguments in the ontology engineering discussion. The arguments exchanged in each of the discussions were classified according to types of arguments proposed in the rhetorical structure theory (*cf.* Section 2.3.3). The dialogs were further evaluated counting the number of agreed issues, judging the clarity of the

discussion and questioning the satisfaction of the participants. Evaluation measures are the number of agreed issues, the clarity of the discussion and user satisfaction. The restriction of arguments indeed increased all three measures.

4.4.4.2. Results from our Experiments

Our experiments provide strong indication that a restriction of possible arguments can enhance the ontology engineering effort in a distributed environment. Referring to the rhetorical structure theory (RST) terminology the arguments elaboration, evaluation & justification, alternatives, example and counter example focus the discussion and convince the participants.

Elaboration An elaboration presents additional detail about the matter of discourse. Possible elaborations are the introduction of members of a set, instantiations of an abstraction, parts of a whole, stages of a process, attributes of an object or specializations of a general issue.

Evaluation & Justification An evaluation provides a measurable advantage for a particular matter of discourse in comparison to another. A justification gives evidence, that something or someone has the authority to make a statement.

Alternative An alternative is a comparable solution for the matter of discourse.

Example An example[11] for a particular matter of discourse increases the belief in the particular issue or idea.

Counter Example A counter example provides counter evidence for a particular matter of discourse and decreases the belief in a particular issue or idea.

4.4.4.3. Example

The following discussion transcript is a part of an experiment performed at our institute (*cf.* Section 5.4.2). The participants have been asked to build an ontology modeling the research interests of our group. The participants have provided their arguments in free text without using the *Argumentation Ontology*. The annotation shown in the example has been added afterwards.

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

...

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)

[11] In RST also called evidence.

Formalization
Individual(issue1 type*(Issue)* value*(raisedIssue "I suggest DKM"))*
Individual(issue1 type*(Issue)* value*(givenBy actorCS))*
Individual(justi1 type*(Justification)* value*(hasArgumentation argumentation1))*
Individual(justi1 type*(Justification)* value*(argumentsOn issue1))*
Individual(argumentation1 type*(HumanArgumentation)* value*(providesText "We have ... lately"))*
Individual(idea1 type*(Idea)* value*(responsesToIssue issue1))*
Individual(idea1 type*(Idea)* value*(ontoChange "add(DKM:Topic)"))*
Individual(elaboration2 type*(Elaboration)* value*(raisedIssue "P2P subtopic DKM"))*
Individual(idea2 type*(Idea)* value*(responsesToIssue elaboration2))*
Individual(idea2 type*(Idea)* value*(ontoChange "add(DKM supertopic P2P)"))*

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".

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

Formalization
...
Individual(idea3 type*(Idea)* value*(ontoChange "add(KM:Concept)"))*
...

A third actor agrees also implicitly, that "P2P" and "DKM" are important for the domain, but challenges that they should be modeled in the proposed way.

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

Formalization
Individual(counter1 type*(CounterExample)* value*(hasArgumentation argumentation3))*
Individual(counter1 type*(CounterExample)* value*(argumentsOn elaboration2))*
Individual(argumentation2 type*(HumanArgumentation)* value*(providesText "File exchange ... KM"))*

The fourth actor presents a new issue which may resolve the conflict.

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

Formalization
...
Individual(issue2 type*(Issue)* value*(raisedIssue "I suggest DC"))*
Individual(elaboration3 type*(Elaboration)* value*(... ...))*
...

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.

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

Formalization
Individual(position1 type*(Agree)* value*(positionOn elaboration3))*
...

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.

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

This example demonstrates that an OE discussion can be modeled with the DILIGENT *Argumentation Ontology*.

4.5. DILIGENT Tool Support

The efficient application of a process model depends on the tools supporting its stages. The following section examines each stage in order to derive requirements for such tools. The case study experiences and literature analysis are the main drivers of this analysis, thus in other application scenarios new requirements may emerge. Based on the process specific requirements a plug-in for the OntoEdit ontology engineering environment (OEE) has been implemented. The argumentation related requirements are realized using standard wiki and online chat tools.

4.5.1. Requirements Derived from the Process Stages

Grouped by the process stages we describe for each activity the process specific requirements. In case an activity duplicates the requirements of another activity we reference the ones mentioned earlier. Section 4.5.2 maps the derived requirements on the tools, therefore we enumerate the requirements and highlight their names like (**requirement (0)**).

4.5.1.1. Central Build

Available methodologies propose different ways to specify the domain (**Domain analysis (1)**), *e.g.*, domain expert interviews, literature studies or workshops. Depending on the elicitation procedure results of these activities are, *e.g.*, transcripts, lists of documents, lists of competency questions or mind maps. 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 ontologies to be reused, the competency questions and other relevant meta information (*cf.* Section 4.3.1.5).

After the specification of the requirements 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 (2)**). The integration of reused ontologies should be supported by, *e.g.*, import features for different ontology representation languages, merging, matching and pruning functionalities.

The board should evaluate the shared ontology according to predefined metrics (**Ontology evaluation (3)**), such as ensuring the validity of the ontology w.r.t. the chosen representation language. Integrated feedback mechanisms can help to capture user satisfaction with the ontology. The tool should support ontology evaluation.

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 on tools supporting arbitrary argumentation (Conklin & Begeman, 1988) we identified the following requirements. These requirements are later on referred to as (**Argument provision (4)**).

1. (**Communication (4.1)**) In the DILIGENT scenario participants of the ontology engineering discussion are locally distributed and they may discuss online. This means that all participants should be able to obtain up-to-date information about the exchanged arguments, the agreed issues, the issues currently under discussion and the participants. The engineering process may also be organized asynchronously. In this case the discussants provide their arguments independently of each other. The tool should highlight new information and inform participants if someone opposes to or agrees with their suggestions.

2. (**Issue / Idea stack (4.2)**) 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 (4.3)**) 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 (4.4)**) If an issue cannot be agreed upon unanimously, the participants may vote for their favorite solution. The tool should support different decision mechanisms, such as majority voting. The participants should be given a time frame to vote.

5. (**Integration into OEE (4.5)**) The integration of the argumentation with an ontology editor is important, because the discussants should not change their working environment for discussing the ontology while building it. Although argument capturing has immediate and long term benefits, it requires additional effort from the users. Usability aspects are therefore of particular interest. It has the further advantage, that reasoning capabilities and different visualization techniques are already available.

6. (**Concurrent ontology visualization (4.6)**) During the discussion the ontology evolves in different ways. While some parts are already agreed by all participants others are still under discussion and thus different models of the same ontology exit. 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 (4.7)**) 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 (5)**) 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 ontology reuse scenarios, it should come in different languages and include links to the argumentation underlying the modeling decisions.

4.5.1.2. Local Adaptation

In order to select the requirements motivated by the activities in the local adaptation stage we distinguish two main types of users. The less frequent type is the user with ontology engineering competence who analyzes his personal needs, conceptualizes and formalizes them. He uses established ontological guidelines in order to maintain soundness and validity of the local ontology (Guarino & Welty, 2002). Besides, he annotates his knowledge according to his locally extended ontology.

The more common type of user is not aware of the exact implications of a formal ontology. The shared ontology is regarded as a predefined categorization schema, such as the categorization he defines in his daily work (*e.g.*, his folder structures). Annotations of documents are primarily provided in the form of assignments to a specific category.

The analysis of the shared ontology can be facilitated by an appropriate visualization (**Visualization (6)**). A detailed requirements analysis for ontology visualization is beyond the scope of this thesis. In DILIGENT 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 concepts most relevant to the user context may be useful.
However, interaction patterns with the ontology depend on the use case for the ontology (**Ontology use (7)**). Important aspects are a domain specific query interface and answer visualization.

Some users organize their local information according to the ontology, thus they need to access it (**Access to local information (8)**). The general solution to link a source with an repository is the introduction of mediators (Wiederhold, 1992; Abecker & van Elst, 2004), which extract information from the sources and transform it in the target representation language. In order to support later stages in our process we require that provenance information is stored. Another aspect regards the annotation of the local information according to the ontology (**Annotation (9)**). 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 (Handschuh, 2005) 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 are captured following the argumentation model (**Argument provision (4)**). 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 are met by locally editing the shared ontology or by consulting and integrating ontologies from other users (**Ontology editing (2)**). Automatic generation of an ontology based on locally available information can facilitate the former strategy (**Ontology learning (10)**). A prerequisite to follow the latter strategy is the availability and accessibility of remote ontologies (**Access to remote ontologies (11)**). Although in our case studies all participants represented their ontologies in the same representation language translation support is required

in other cases. The assessment , including documentation analysis, argument analysis, concept analysis, of the retrieved ontologies (**Visualization (6)**) is followed by their local integration, including entity selection, entity mapping, entity alignment (**Ontology integration (12)**), and provenance storing (**Provenance (13)**). The evaluation of the adapted ontology is performed according to the predefined measures (**Ontology evaluation (3)**) and documented to facilitate future usage (**Documentation (5)**).

4.5.1.3. Central Analysis

The analysis of the user requests is preceded by the collection of information (**Access to remote ontologies (11)**). The board may trigger the information collection more frequently then they have joint meetings, as they may gather only if a certain number of changes was 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 parts of the local update stage 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 (6)**) and group it according to different measures (**Clustering (14)**). Experience from our case studies suggest that the analysis can start with a simple alphabetical sorting of the introduced ontology entities. However, different groupings according to similar topics or similar aspects of the ontology are desirable. 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 (15)**), 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.

4.5.1.4. Central Revision

The conceptualizations following from requirements identified in the previous stage are implemented in the central revision stage. This requires basic editing functionality (**Ontology editing (2)**). The ontology changes are resolved taking into account that the consistency of the underlying ontology and all dependent artifacts are preserved (Stojanovic et al., 2002).

The integration of the new shared ontology with the user local ontologies should be possible (**Provenance (13)**). The changes to the shared ontology based on direct integration of user proposals should reference the original request. This is also required for changes

which are motivated by user changes or user arguments. In this case the provenance information should include all arguments and changes which lead to the specific decision.

The final changed shared ontology is made available to the users (**Publishing (16)**). Either it is published centrally requiring the users to obtain it on their own or it is sent to to them directly.

4.5.1.5. 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 analyze the new version of the shared ontology they need different visualization options (**Visualization (6)**). 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 requests are of particular interest to him. The user requests can be integrated into the shared ontology, either directly, conceptually or based on arguments. The evaluation of the control measurements (*cf.* 4.3.3 and 4.3.5) can help the user making his decision (**Measurement calculation (15)**). 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 should 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 propagated to the local ontology (**Ontology integration (12)**). 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. The user should be enabled to use the shared model instead of his own adaptations, if those adaptations have been integrated in the shared model. Furthermore, the board may have included a change based on user arguments, but has drawn different conclusions. Here the user should decide whether he prefers the shared interpretation or his own. If the user realizes, however, that his decision to update has been a mistake a possibility to restore the old status is required.

Table 4.4 summarizes the tool requirements and lists the tools addressing them. The next Sections elaborate on the tools.

4.5.2. DILIGENT OntoEdit Plugin

In the previous section we focused on the technical requirements resulting from the DILIGENT process. The general requirements are realized in the SWAP system and are tailored around the specific needs of our case study partners (*cf.* Section 3.1)[12]. For this purposes

[12]In other application scenarios tool support is influenced by the specific needs of the use cases.

the SWAP system integrates an OEE implementing the necessary functionality following the approach proposed by (Sure, 2003): methodological support functions are added to an existing OEE building on the available standard functionalities and adding methodology specific methods with plug-ins. The OntoEdit OEE plug-in framework is used in this case (Sure et al., 2003). The selection of the specific tool was arbitrary as many established OEEs offer plug-in functionality (*cf. e.g.*, (Maedche et al., 2003; Noy et al., 2001; Kalyanpur et al., 2005)). A detailed description of the plug-in infrastructure can be found in Handschuh (2005). We, however, integrated functionalities into the OEE only as far as they were essential to the ontology engineering process, for some, such as access to local information sources, independent tools are used. The following section enumerates the functionalities of the OntoEdit plug-in and related tools and aligns them with the requirements.

Requirement	No.	Tool	Section
Domain analysis	1	OntoEdit	4.5.2.1
Ontology editing	2	OntoEdit	4.5.2.1
Ontology evaluation	3	OntoEdit	4.5.2.1
Argument provision	4	wiki, chat	4.5.3
Communication	4.1	wiki, chat	4.5.3.1
Issue / Idea stack	4.2	wiki, chat	4.5.3.2
Classifiable arguments	4.3	wiki	4.5.3.3
Voting mechanism	4.4	wiki, chat	4.5.3.4
Integration into OEE	4.5	manual	4.5.3.5
Concurrent ontology visualization	4.6	manual	4.5.3.6
Moderator	4.7	manual	4.5.3.7
Documentation	5	OntoEdit	4.5.2.1
Visualization	6	OntoEdit	4.5.2.2
Ontology use	7	SWAP system (User Interface)	3.3
Access to local information	8	OntoScrape (Extractor)	3.3 & 4.5.2.3
Annotation	9	SWAP system (OntoScrape)	4.5.2.4
Ontology learning	10	Prototyp	4.5.2.5
Access to remote ontologies	11	SWAP system (LNR, Communication Adapter)	3.3 & 4.5.2.6
Ontology integration	12	OntoEdit	4.5.2.7
Provenance	13	SWAP metadata model	3.4.1 & 4.5.2.8
Clustering	14	OntoEdit plug-in	4.5.2.9
Measurement calculation	15	OntoEdit plug-in	4.5.2.10
Publishing	16	email, website	4.5.2.11

Table 4.4.: List of Tool Requirements for DILIGENT

Running example In order to illustrate the different functionalities implemented in the XAROP application to support the DILIGENT process, we introduce an example. In Figure 4.9 the two peers, *Goethe* and *Schiller* are visualized (*cf.* Frame A)[13]. Both peers share folders: *Goethe* contributes information stored in folder "share1" (*cf.* Frame B) and *Schiller* contributes information stored in folder "share2" (*cf.* Frame C). *Schiller* has extended the shared ontology reusing his local folder structure: The concept Document in the shared ontology is now superclass to the 100. Projektorga and its subclasses, respectively subfolders (*cf.* Frame D). We observe in *Schiller*'s clustermap pane that the folder "140. Finanzen" contains no files (`schiller: 140. Finanzen (0)`) (*cf.* Frame E), while 24 files are assigned to the concept 140. Finanzen (`schiller: 140. Finanzen (24)`) (*cf.* Frame F). The files related to the subclasses of 140. Finanzen, *e.g.*, Long Cost statement 200209, are also related to the class 140. Finanzen, because of the inheritance implied by the rdfs:subClassOf relationship. *Goethe* consulted *Schiller* and adapted the shared concept Document by reusing *Schiller*'s extensions (*cf.* Frame G). *Goethe* retrieved the files related to the concept 140. Finanzen from *Schiller* (*cf.* Frame H), since we observe that he himself does not share any information related to that concept ((goethe: 140. Finanzen (0))) (*cf.* Frame I).

4.5.2.1. Domain Analysis, Ontology Editing, Ontology Evaluation and Documentation

OntoEdit is an ontology engineering environment providing basic ontology management services, such as ontology editing, ontology documentation, ontology visualization, and storage (Sure et al., 2002; Sure et al., 2003). The open plug-in framework enables the integration of a number of extensions, *e.g.*, plug-ins are available for domain analysis, ontology translation, and competency question capturing and ontology evaluation (Sure, 2003). In particular OntoEdit offers advanced support for collaboration and integration of the inferencing capabilities. A re-implementation of OntoEdit based on the Eclipse[14] framework has recently been made available under the new name OntoStudio[15].

4.5.2.2. Visualization

OntoEdit integrates a number of visualization paradigms. It can, however, visualize only complete ontologies and has no view selection mechanism. As in our scenario the number of newly created entities within the P2P network can be large, the views can be generated using queries. Queries can be defined manually, or predefined ones — visualizing certain branches of the ontology — can be selected (Figure 4.10). SeRQL offers the mechanisms to realize this option.

[13] The letters refer to the labeled frames in Figure 4.9. Frames with the same label visualize related contents under different visualization paradigms

[14] `http://www.eclipse.org/`

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

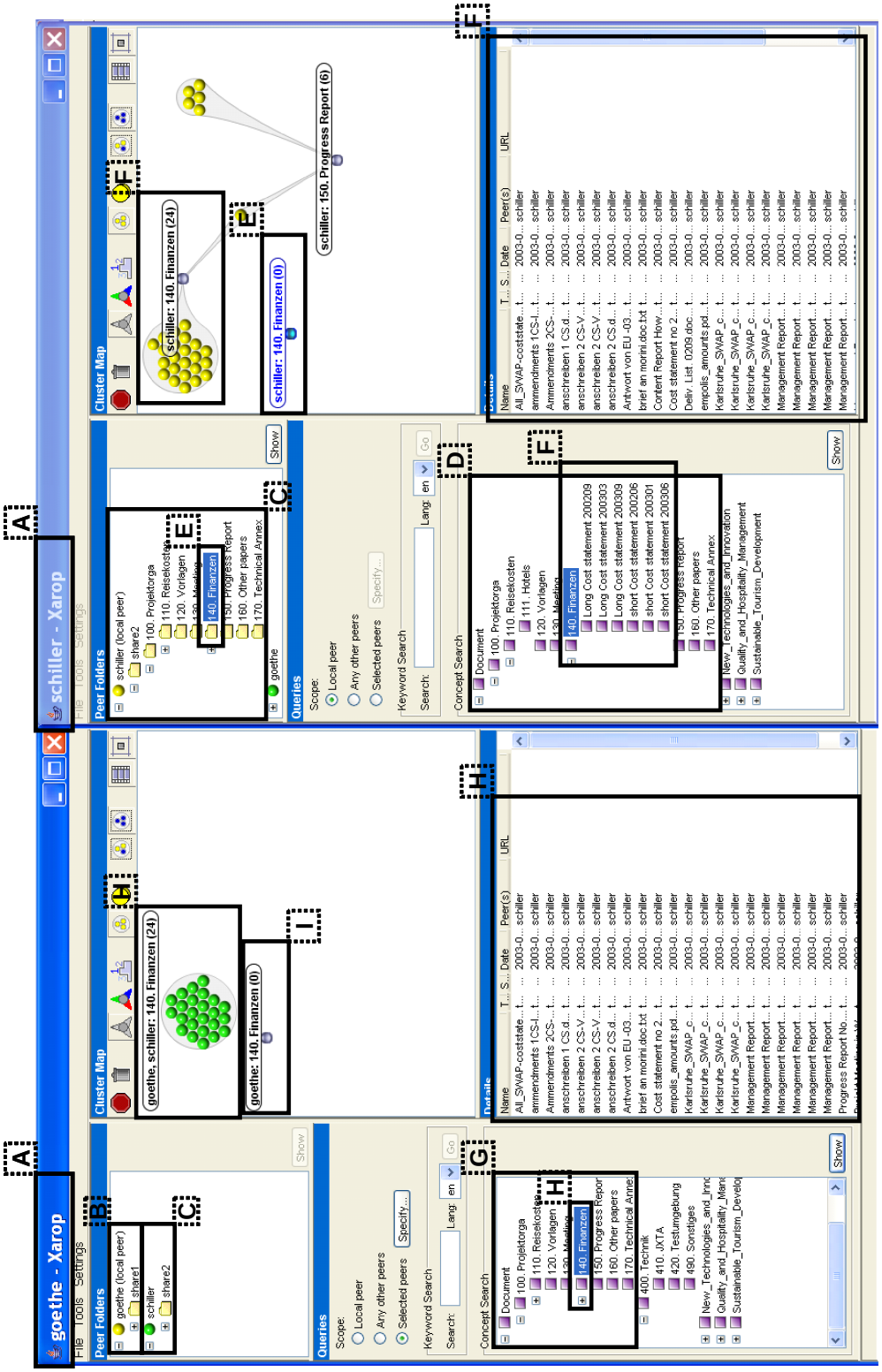


Figure 4.9.: XAROP Application

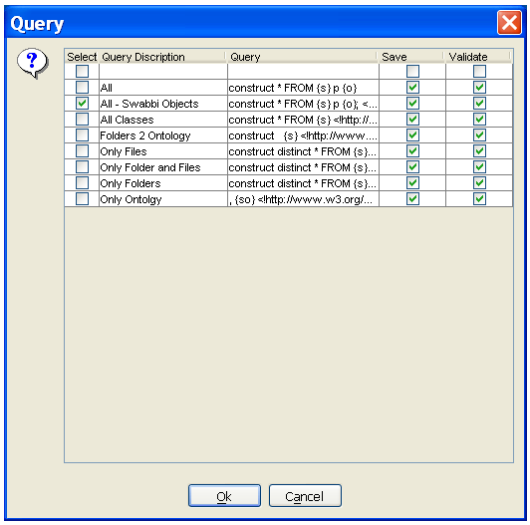


Figure 4.10.: Personalized Views on Local Ontologies

In order to separate extensions made by different users and to distinguish their relative activity we change the background color of the entities depending on their source. Since each peer uses its own name space to create URIs, extensions to the core made by different peers can be distinguished. The tool highlights the concepts, relations and instances of different peers by changing their background color. The saturation and brightness of the color indicates the number of concepts coming from a particular peer.[16] White is preserved for name spaces which the users can choose not to highlight (*e.g.*, the local, swap-peer and swap-common name space are excluded from highlighting by default).

4.5.2.3. Access to Local Information

The OntoScrape tool is an implementation of the Extractor component introduced in Section 3.3; it can extract information from the user local file and email system. OntoScrape extracts file system information, such as the folder hierarchy, bookmarks, and emails and builds up an RDF(S) representation in which the folder names are used to create instances of class Folder. As an implementation of the Editor component the DILIGENT OntoEdit plug-in provides access to the LNR.

4.5.2.4. Annotation

In our case studies documents play a predominant role for the population of the ontology. OntoScrape generates automatically instances of concept Source, *e.g.*, Folder and File. In

[16] Brighter and less saturated means less concepts than darker and more saturated.

order to relate files with concepts of the ontology a drag and drop option is implemented in the XAROP interface. If folders are reused to extend the shared ontology the respective files are related to the created concept.

Outlook Automatic text classification techniques, nowadays very effective, can detect relations between files and concepts in the ontology. The XAROP application includes an interface for classifiers to suggest document classifications. Classifier training can take place remotely for the core ontology according to established procedures (Sebastiani, 2002). The classifier has to produce a set of RDFS statements, stating which files should be classified where in the concept hierarchy.

Annotations going beyond classifications require more detailed analysis of the documents at hand. Well known techniques from ontology annotation can be integrated (Handschuh, 2005).

4.5.2.5. Ontology Learning

The manual integration of locally available information with the shared ontology was sufficient in our case study. Nevertheless, as the amount of shared information and the size of the shared ontology grow, the process requires automated support. Research in *Ontology learning* focuses on methods and tools to automatically create ontologies from available information in particular texts (Maedche & Staab, 2001). Creation of ontologies from classification structures, such as folders, however, has not received much attention. Inspired by a general ontology learning process (Maedche & Staab, 2001), for the particular case Lamparter et al. (2004) develops a process model and a prototypical implementation. Although the detailed description of the developed algorithms is out of the scope of this thesis, the example in Figure 4.11 demonstrates the idea and shows the feasibility of the approach to extract ontologies from local structures. The algorithms take as input the folder hierarchy from the local user, start with the extractions of words from folder labels and try to identify candidates for concepts. All words which are discarded from the list of concept candidates become members of the instance candidate list. The identification of taxonomic relationships between the candidate concepts precedes the introduction of other relationships. Finally, the algorithm tries to align the instances with the extracted concepts.

The algorithms incorporate knowledge from electronic lexical reference systems, such as WordNet[17], and available ontologies, such as the shared ontology, to classify labels as concepts, instances or relations. The algorithms utilize co-occurring labels, the sub-folder hierarchy and named-entity recognition as further information. The performance of the algorithms was evaluated comparing their output with hand-build ontologies. Depending on the quality of the input folder structures measured in terms of usage of complete words for folder labels, the depth of the hierarchy, and the communality with the shared ontology the

[17] `http://wordnet.princeton.edu/`

algorithms achieve acceptable results in terms of recall and precision recognizing between 70% and 80% of the human gold standard.

In Figure 4.11 the algorithms identify in the first step the folder labels Conferences, Papers and Presentations and EU-Projects as candidate concepts from a comparison with WordNet and a shared ontology. According to the available ontologies the algorithms adds concepts for Project, Deliverable to the list of concept candidates and classifies EU-Project, Paper, Presentation and Conference as concepts. The algorithms recognize 'Cyprus', 'SEKT', 'ODBASE', 'OntoMapping.pdf' and '2004' as named-entities. The instance to class assignment is based on the relation ships found in the folder taxonomy.

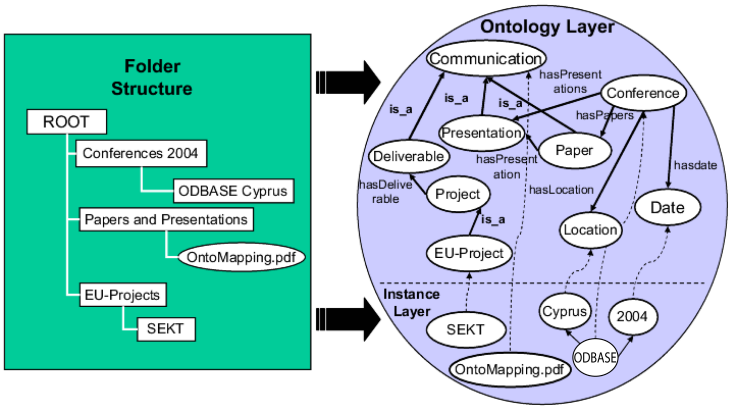


Figure 4.11.: Semi-automatic Ontology Creation

4.5.2.6. Access to Remote Ontologies

A user can retrieve ontologies from other participants in order to reuse their conceptualizations locally. Alternatively, he may use the query result only for inspiration and create own extensions and modifications. We have realized this in the XAROP application, and we have introduced a new option to OntoEdit. We submit a predefined SERQL query, exchanging a placeholder with the selected concept, to the selected peers and search for concepts and properties related to the selected concept. In our example *Goethe* consults *Schiller* for his extensions of the concept Document as visualized in Frame G1 in Figure 4.12 and Frame G2 in Figure 4.15.

4.5.2.7. Ontology Integration

Manual integration of local structures In our case study users reused their local folder structures to adapt the shared ontology. Our plug-in added a new option, allowing the selection of a set of instances of Folder in order to create concepts or relations using the selected folder names. The folder hierarchy given by the inFolder relation can be used to

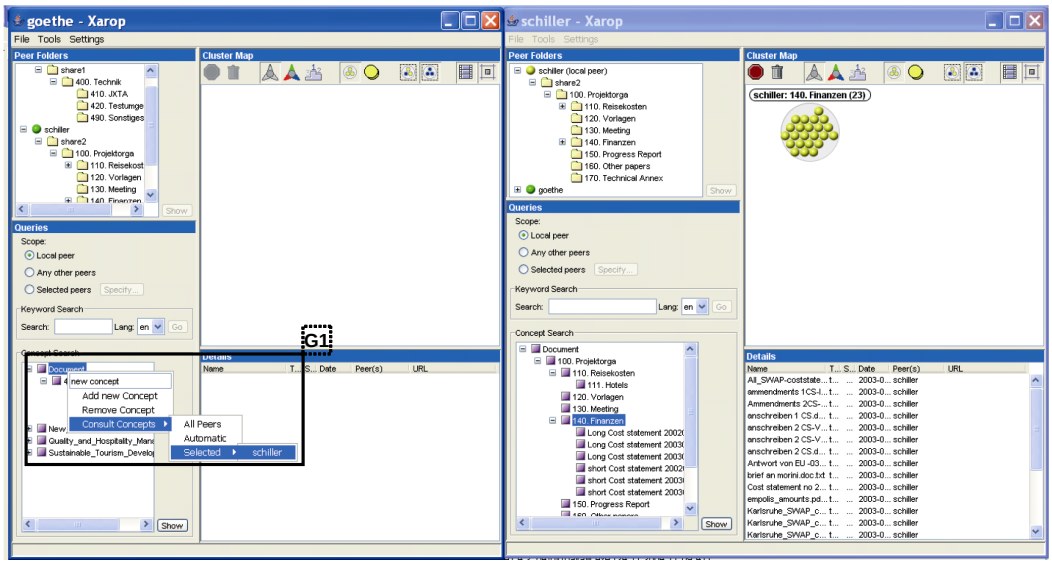


Figure 4.12.: Access to Remote Ontologies

construct a subclass-of hierarchy. Besides the simplification of the ontology construction, an advantage of this technique is the implied traceability; the origins of a concept created in this way can be traced back to the original folder (*cf.* Frame N2 in Figure 4.15). Additionally all information stored in the original folder is still accessible through the concept. Although this created some conflicts w.r.t. the ontological interpretation of the information concerned, it proved very useful from a user perspective[18].

Not all users wanted to use OntoEdit to change the shared ontology. Therefore, we have added a simplified version of the afore mentioned option to the search interface. The user can drag folders in the folder view and drop them in the concept view to create new concepts as visualized in Figure 4.13. In our example *Goethe* has extended the shared concept Document reusing the subfolders of folder "400. Technik" (*cf.* Frame J).

Alignment SWAPSTER integrates a component for semi-automatic alignment. Alignment detection is based on similarities between concepts and relations(*cf. e.g.*, (Noy & Musen, 2002)). The user may either select a set of classes and ask for proposed alignment for these classes, or he can look for alignments for the entire class hierarchy. The reader may note that even the best available alignment methods are not accurate and hence the alignment process requires user involvement. For a detailed description of this tool, the reader is referred to (Ehrig, 2006), as it is beyond the scope of this work.

[18] It is very difficult to determine the exact relationship between, *e.g.*, a file containing information about a call for papers for a specific conference stored in a folder *conferences* and the respective concept conferences.

Continuing our example, *Goethe* does now also share his own folder "100. Projectorga" and related subfolders. Mapping detection is time consuming, because the naive approach requires a quadratic number of comparisons in the number entities to compare. Although the algorithm presented in Ehrig (2006) is more efficient than the naive approach a complete run comparing large ontologies may still overstretch the users patients. Therefore, the algorithm provides continues updates, and the user can analyze the proposals immediately. One step in this update procedure is visualized in figures 4.14(a) and 4.14(b). In the second figure more mappings have been detected. The tools input is *Goethe*'s local information. As *Goethe*'s ontology is in a different namespace than *Schiller*'s the algorithm is able to separate the two and to suggest possible alignments. The alignment algorithm is accessible as part of the OntoEdit plug-in as visualized in Figure 4.15 Frame K2.

Outlook The integration of ontologies from different sources can lead to inconsistencies on the conceptual, the formal or both levels. Depending on the semantics of the formalization language, there are technical solutions to resolve formal inconsistencies *cf.* (Maedche et al., 2003). The integration of the new shared ontology with the local ontology, however, remains a challenging technical problem yet to be solved.

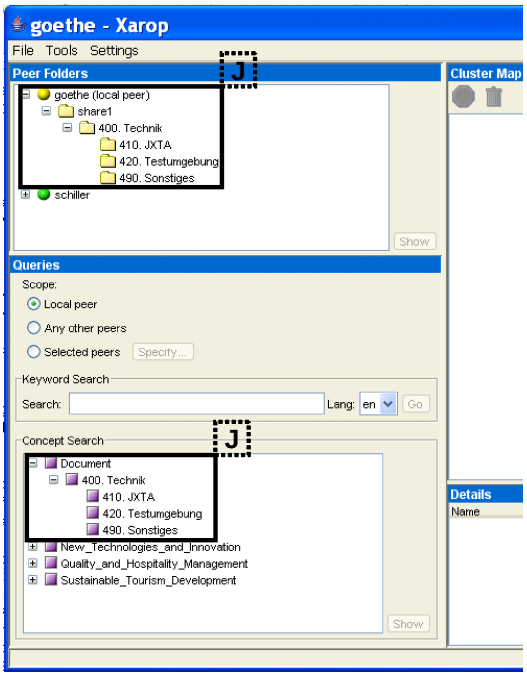
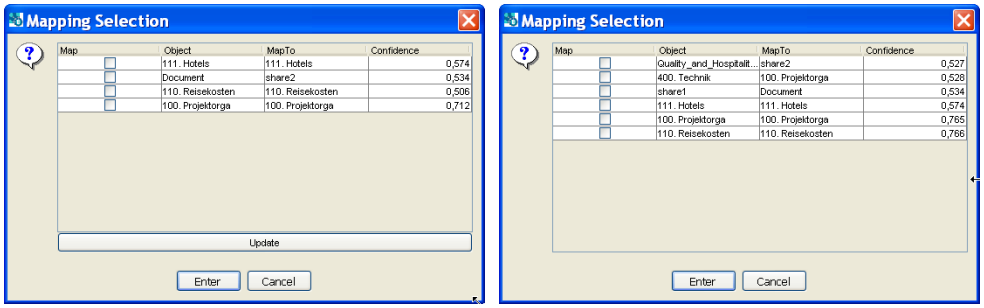


Figure 4.13.: Local Customization of the Local Ontology



(a) First Result (b) Final Result

Figure 4.14.: Ontology Alignment Support

4.5.2.8. Provenance

As elaborated in the requirements analysis we need provenance information in two stages of the DILIGENT process. In order to detect user changes in the analysis stage the board needs knowledge about their origins. In order to update to a new version of the shared ontology in the local update stage the user needs the reasons for the introduced changes. In the current version of the XAROP application we support the first use case. For that case the SWAP metadata model provides the infrastructure to store provenance information (*cf.* Section 3.4, page 40). All ontology entities transmitted through the XAROP application can be traced to their origins.

In the example two Swabbi-objects are available for the concept Document, because *Goethe* requested all sub-concepts of that class from *Schiller* (*cf.* Figure 4.15). The new option `ShowSwabbi` allows to access all meta information stored in the Swabbi-object created for a selected ontology entity (*cf.* Frame M2 in Figure 4.15).

4.5.2.9. Clustering

Clustering refers to the method of grouping objects with similar properties close together while separating objects with different properties. The main challenge is the identification of the properties which are relevant for a given task. As the total number of ontology entities was tractable without sophisticated clustering methods it was sufficient to use the labels, the occurrence and the peer activity as properties to group the ontology entities. This allowed the user to recognize concepts with the same label coming from different remote local ontologies and to identify concepts which were reused by a large number of peers. In Figure 4.15 Frame L2 highlights the different sorting possibilities.

4.5.2.10. Measurement Calculation

The evaluation measures defined as criteria in the controlling activities can be calculated from the information stored in the SWAP data model. For our case studies it was sufficient to calculate the adaptation rate, which indicates how many user have included the concept into their local ontology. It is defined according to Equation 4.1. The adaptation rate is visualized as a tool tip in OntoEdit (*cf.* Figure 4.15 Frame O2).

4.5.2.11. Publishing

In the case studies the shared ontology was published on a Web site and the users had to retrieve it themselves in order to update to the new version. The initial shared ontology is

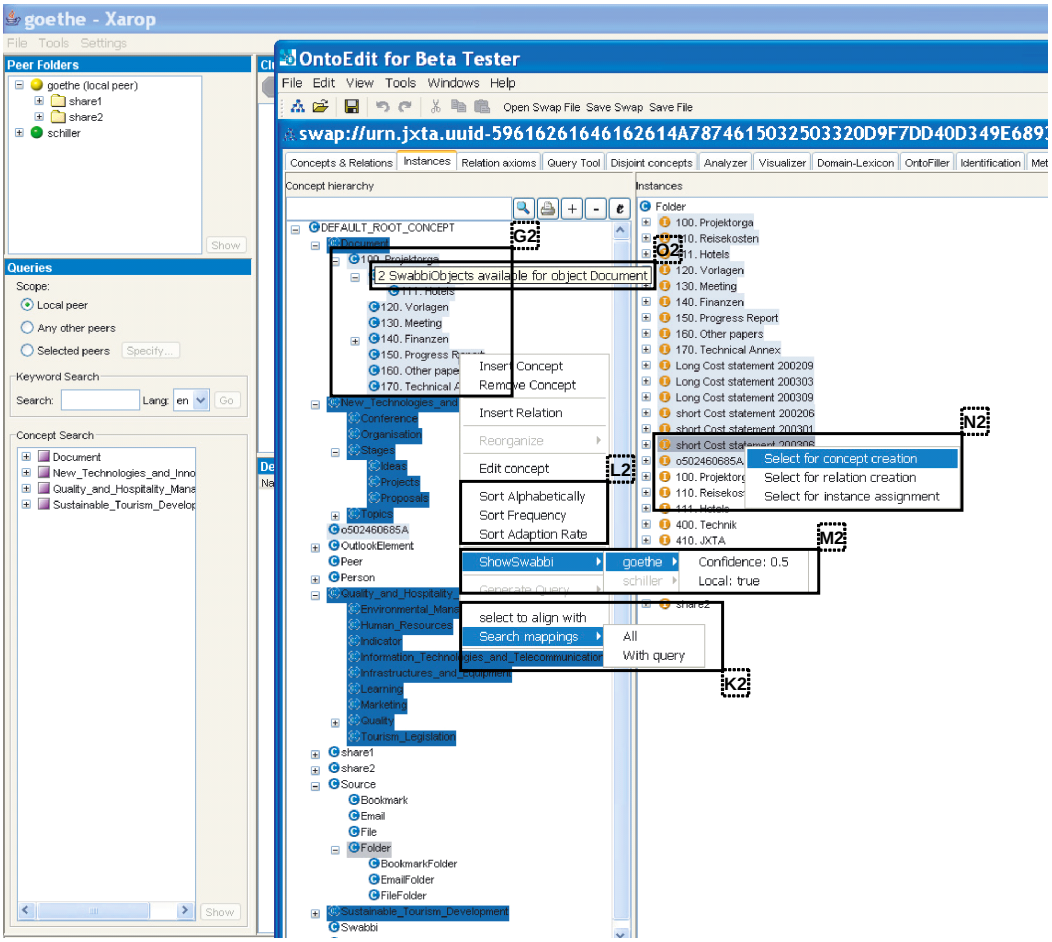


Figure 4.15.: DILIGENT OntoEdit Plug-in

integrated into the main distribution of the application.

Outlook The advertisement component can transmit a list of changes to the users in future versions of the XAROP application. Maedche et al. (2003) describes an infrastructure to enable consistent change propagation in a distributed environment.

4.5.3. DILIGENT Argumentation Tools

The DILIGENT plug-in, integrated into the XAROP application, supported the DILIGENT stages in the IBIT case studies as described in Section 5.4.1. The case study emphasized the advantages of an argumentation model to provide more structured guidance to ontology engineering discussions. The argumentation model itself, however, could not be applied in that case study. The developed tools were conceived with the requirements of the AIFB case study (*cf.* Section 5.4.2) and the legal case study (*cf.* Section 5.4.3) in mind. In order to demonstrate the feasibility of the approach to ontology engineering, we adapted existing technology and used it for evaluation purposes. In the following we discuss the two tools, w.r.t. the requirements listed in Section 4.5.1.

Argument Provision

AIFB In the AIFB case study we used two chat clients, allowing online communication between the participants, and a Web site, visualizing the shared ontology. Discussions about all issues is performed in the same chat client.

Legal In the legal case study we use a wiki[19]. For each ontology entity a new wiki page is created, and the argumentation is captured there.
In Figure 4.16 we see a screenshot of the wiki, running on the SEKT portal, showing the concept Hecho[20] with its description, argumentation and attributes, as well as a graph made with the KAON OIModeller showing the concept and its attributes.

4.5.3.1. Communication

AIFB Instant communication is possible with the chat clients. All participants meet in one virtual chat room. The discussion progress can be stored in a log file.

Legal Wiki pages are accessible via standard Web browsers. In our case, any user can edit a wiki page. The communication is not instantious, since only one user can edit a wiki page at a time. Different users, however, can edit other wiki pages simultaneously.

[19] A wiki is a Web application that allows users to add content, as on an Internet forum, but also allows anyone to edit the content. (Wikipedia, 2005)

[20] *Translation:* Act

4.5.3.2. Issue/Idea stack

AIFB Issues are raised in one chat client, while the discussion and idea proposal takes place in the other chat client. The moderator keeps track of the raised issues, orders them and ensures that only one is discussed at a time.

Legal One wiki page is reserved to raise issues. Each idea is discussed on a distinct wiki page.

4.5.3.3. Classifiable arguments

AIFB The chat clients allow arbitrary communication, thus users may provide all kinds of arguments. The classification is done manual after the discussion. The moderator asks the participants to reformulate statements, if they do not match the predefined argument types. The moderator enforces the use of one chat client for discussions and one for hand rising and voting.

Legal As a chat client the wiki allows arbitrary communication. The classification is done manually during the discussion. The wiki provides specific tags to state the

Figure 4.16.: Wiki-based Argumentation

type of argument provided.

4.5.3.4. Voting mechanism

AIFB One chat client is reserved for voting and issue raising. The moderator counts manually the votes. Decisions are based on a majority vote for an idea.

Legal Votes are tracked on the wiki pages, and tagged as such. The moderator counts the votes manually for the ideas and decides based on the majority principle.

4.5.3.5. Integration into Ontology Editor

The tools are not directly integrated with an ontology editor. Users who want to visualize and show the created ontology to remote users, need to manully integrate the changes and publish an image of the new ontology.

4.5.3.6. Concurrent ontology visualization

AIFB The moderator models the shared ontology in an standard ontology editor. Different versions of the ontology are modeled separately. Screenshots of the ontology versions are published as a screen-shot on a Web page.

Legal As in the previous case, the ontology is modeled in standard ontology editors; images of the current agreed version are published in the wiki.

4.5.3.7. Moderator

In both cases no special support for the moderator is available.

Outlook Based on our past experiences we have developed a screen design and interaction patterns for a plug-in meeting all requirements for argument provision. The plug-in is designed for the OntoStudio OEE, and will be implemented in the SEKT project (Vrandecic et al., 2006).

4.6. Summary and Outlook

This chapter develops the ontology engineering methodology DILIGENT for distributed knowledge management scenarios. In comparison to ontology development for centralized knowledge management applications the distributed setting requires support for autonomously acting users. They are distributed across many places and may not have on-

tology engineering background. The ontology they build is evolving in order to meet new needs.

DILIGENT addresses the new requirements in a twofold manner. First, it proposes a process model in which a shared ontology is distributed to the users and locally changed and a board continuously updates the shared ontology based on these changes. Second, an argumentation framework facilitates the externalization of assumptions underlying an ontology. The framework proposes a process for argument provision and a semi-formal model to provide a restricted set of argument types.
Some of the DILIGENT activities may be supported by tools. This chapter elaborates on these requirements and presents prototypical implementations of tools supporting the process.

The requirements of the distributed knowledge management scenario are very similar to the ones found for ontology engineering in the Semantic Web (Pinto & Martins, 2002). In the vision of the Semantic Web today's Web sites will be semantically enhanced with ontologies in order to exchange information. The Web site owners are autonomous, locally dispersed and predominantly non-expert ontology builders. As in the current Web, offered information will change quickly and the underlying ontologies should be updated regularly. DILIGENT addresses these requirements and should thus be applicable in this broader setting, too.

The need for explicitly stating the assumptions underlying the design of an artifact is not unique for ontology engineering but has been recognized in the field of requirements and software engineering for a long time. The IBIS methodology has been designed for that purpose. Arguments have proved their value to augment the reasoning in the design process, to facilitate software reuse and to help non-participants to understand the design decisions (*cf. e.g.*, (Gotel & Finkelstein, 1994; Buckingham Shum & Hammond, 1994)). Argument capturing may, however, hinder an effective design process due to its overhead. It remains an open research question whether the findings presented in this thesis are of use for those communities.

The application of the DILIGENT methodology in three different case studies is the subject of the next chapter.

5. Evaluation of the DILIGENT Methodology

Is not all honourable work also useful and good?[1]
— Plato, Protagoras, 358b


Overview

The evaluation of the DILIGENT methodology is twofold: first it is compared with other methodologies and second it was applied in three different case studies. The evaluation demonstrates that DILIGENT extends existing ontology engineering methodologies so that ontologies for distributed knowledge management applications can be build and that DILIGENT can be applied in such scenarios.

In the first case study the process model was applied and the activities were validated. The second case study focuses on the selection of relevant arguments for ontology engineering processes. In the third case study the formal argumentation model is used. Finally lessons learned are discussed and future directions for the methodology are proposed.

References: This chapter is based on the publications (Tempich et al., 2004a), (Pinto et al., 2004a), (Pinto et al., 2004b), (Pinto et al., 2005), (Vrandecic et al., 2005), (Tempich et al., 2006), (Sure et al., 2004) and (Casanovas et al., 2005; Tempich et al., 2005c).

5.1. Evaluating a Methodology

> *"Evaluation is a process to compare different approaches to solve a certain problem."*
>
> *(House, 1980)*

An evaluation is always targeted at a specific audience. In particular when it comes to the evaluation of processes, two types of audiences can be distinguished: the economists and

[1] Kai to kalon ergon agathon te kai ophelimon.

managers are interested in numbers and facts; the practitioners are interested in experiences in the application of the process. For that reason different approaches to evaluate processes have been defined. The remainder of this section presents three complementary evaluation approaches targeted at economist as well as at practitioners.

From an epistemological point of view there are objectivist and subjectivist approaches to evaluation. The former 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 of the persons involved in the process. 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 should trust the procedures of the evaluator. The evaluation should thus be "true, credible, and right" (House, 1980, p. 250).

The way the evaluator claims credibility differs depending on the approach to evaluation. The evaluator should be unbiased, interested in the findings but not favor any of the evaluated solutions for the problem. 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. Observing the process under evaluation is the primary method of data collection. Therefore, 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 a process 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 may 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 strengths and weaknesses; following only one evaluation approach may thus be misleading. DILIGENT is therefore evaluated according to more than one model, namely 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. It is thereby evaluated according to the same evaluation measures as other ontology engineering methodologies, such as the OTK methodology or METHONTOLOGY. (i) *Goal free* evaluation aims at the users and consumers and analyzes a process logically to allow for informed *Consumer choice*. We identify strengths and weaknesses of the methodology, thus we compare DILIGENT to other methodologies enabling potential users to decide which methodology to use given certain user requirements. (ii) *Case study* evaluation incorporates the clients and practitioners to *understand the diversity* of the process. The case

study exemplifies the potential experiences following DILIGENT. (iii) In the *Professional review* evaluation experts in the domain of interest examine the model. The designer of the methodology can adjust the model according to the evaluation results.

In the following sections 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.

5.1.1. 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 process which fits the reader best.

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 may 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 criterion the evaluator assigns a value to the evaluated procedures.

Input The basis for a goal free evaluation is an objective set of evaluation criteria for the procedures under evaluation.

Output The goal free evaluation provides the reader with an objective analysis of a set of criteria for a number of procedures.

Advantages The evaluation method is objective, as all models 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 may be the result.

5.1.2. Professional Review

Description The professional review model relies on the experiences of knowledgeable people in the area of interest for evaluation purposes. A number of professionals review a process w.r.t. its plausibility, quality and other criteria. A main feature of the professional review is that the professionals themselves define the relevant evaluation criteria depending on the requirements of the domain. They 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 Established evaluation criteria in the domain may be an input to this model.

Output This evaluation model produces expert opinions regarding the plausibility and understandability of the evaluated process.

Advantages The professional review model reduces bias, since all aspects of a process are evaluated. Hidden consequences of the evaluated process may be detected, as the experienced evaluator inquires into the process details. 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 process.

Disadvantages The outcome of this evaluation depends on the capabilities and opinion of the evaluators. The evaluation depends on the availability of experts and can not regularly be repeated.

5.1.3. 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 Before the evaluation the target audience gets familiar with 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.* (Yin & Campbell, 2003)). 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 and best practices descriptions. The reports enhance understanding rather than offer explanations.

Advantages The main advantages of the case study model is its emphasis on practitioner experiences. It exemplifies for future users which experiences they potentially make, if they follow the evaluated process. Since it incorporates 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 approaches. The view on the process is thus very broad.

Disadvantages The value of the case study description depends on the capabilities of the evaluators. As the evaluator is confronted with many influencing variables it is difficult to extract the meaningful ones. The evaluator or participant should 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 should balance the different viewpoints. This can be best resolved in just portraying the experiences 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.

5.2. Goal Free Evaluation

Table 5.1 compares DILIGENT to other well known methodologies.[2] The OTK methodology (Sure, 2003) and METHONTOLOGY (Gómez-Pérez et al., 2003) have been selected because they are the most complete methodologies to guide ontology development in centralized settings; while HCOME (Kotis & Vouros, 2005) aims at the same scenario as DILIGENT. We have adapted the categorization of (Gómez-Pérez et al., 2003) 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 the different reasons which lead to the conceptualization. These 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 elaborate on support for ontology management and pre-development activities, since these are already well supported by other mature methodologies.

5.3. Professional Review

In order to arrive at the methodology as it was presented in Section 4.3 we have proceeded in several development steps. The main drivers of the development were the results of the case studies as presented in the next section and the feedback from external reviewers. This section presents the development of the DILIGENT methodology as influenced by external reviews, while it distinguishes between the evolution of the process and the development of the argumentation framework.

5.3.1. 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

[2] See Section 8.2.1, page 212 for a more detailed description.

[3] The abbreviations used in this table refer to the level of detail of the respective activity description: *NP*: Not proposed; *Proposed*: Described on the conceptual level; *Described*: Activity is described, but no single tasks; *Descr. in detail*: Detailed description of the activities and related tasks with extended examples.

Feature			METHON-TOLOGY	On-To-Knowledge (OTK)	HCOME	DILIGENT
Ontology management activities	Scheduling		Proposed[3]	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
	■ Argumentation / Decision process		NP	NP	Proposed	Descr. in detail
	Merging and Alignment		NP	NP	Proposed	NP

Table 5.1.: DILIGENT and Related Ontology Engineering Methodologies

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 initial tool support, but did not provide a detailed description of the stages. 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 then further elaborated. They asked for an elaboration on the negotiation process to reach consensus when conflicting changes were submitted to the board. The reviewers criticized the generality of the process description and the brevity of the case study. They questioned 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. The IBIT case study lasted for 3 months. The reviewers underlined the value of DILIGENT for ontology development in the Semantic Web context, 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 may have varying life cycles. The scalability of the process model was of further concern, since it was not clear how much it would cost 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 this version of the methodology we 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 controlling activities with the respective decision metrics have not been completely evaluated yet. Another case study with more users would also require further development of the presented tools. Another open issue relates to the costs of the process. Future research should determine the tradeoffs between the agreement costs for the shared ontology and mapping costs for different ontologies.

5.3.2. 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. The initial framework defined the argumentation process and proposed a selection of argument types based on a literature review. The reviewers recognized the importance for ontology engineering. They suggested the application of the argumentation framework for ontology integration.

We subsequently extended the argumentation framework with the argumentation ontology. The reviewers emphasized the validity of the argumentation framework extension. They highlighted the significance of tool support.

Open issues Although we have not developed specialized tool support for the argumentation framework, the framework 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 (*cf.* (Vrandecic et al., 2006)).

5.4. Case Studies

In this section we describe the application of the DILIGENT methodology in case studies. Each case study focused on particular aspects of the methodology as summarized in Table 5.2. In the IBIT case study we examined the process model, its stages, activities and tasks. The AIFB case study focuses on the argumentation framework and the selection of efficient argument types. The argumentation framework is also the main focus of the legal case study.

For each of the case studies we describe the organizational background and the objectives for the application of DILIGENT. We go into detail with the data collection process and its results. From each case study we have collected a number of lessons learned. We gathered the data from our case studies interviewing its participants and by own observations.

Case study	DILIGENT contribution				
	Decentralization	**Partial Autonomy**	**Iteration**	**Non experts**	**Argumentation**
IBIT case study	Yes	Yes	Yes	Yes	No
AIFB case study	No	No	Yes	No	Yes
Legal case study	No	No	No	Yes	Yes

Table 5.2.: DILIGENT in the Case Studies

5.4.1. The IBIT Case Study

The IBIT case study was part of the SWAP project. Besides the evaluation of the DILIGENT process the case study served as a testbed for the XAROP application.

5.4.1.1. Data Collection Process

The IBIT case study lasted for three month, in which the users followed the DILIGENT process. Each stage of the cycle was performed twice. Data was collected in the beginning, in week six, after the first cycle, and at the end of the case study, after the second cycle. We interviewed the two organizers of the case study and several users of the system in order to collect information about the *Local adaptation* and *Local update* stage. It was evaluated whether the system and the process met their requirements and expectations. The evaluation examined in detail the task execution, its order and if the users performed tasks which are not defined in the process model.

The criteria to evaluate the ontology related to the acceptance of the ontology and the coverage of required knowledge. The usability and the usage of the tools was another aspect of the evaluation. The evaluators participated in the *Central Analysis and Revision* stage.

5.4.1.2. Objectives for the Case Study

In the IBIT case study we deployed the XAROP system to enable information sharing between the participants and had two objectives. One the one hand we wanted to evaluate XAROP from a technical point of view. On the other hand we aimed at an evaluation of DILIGENT. Regarding the methodology we had the hypothesis, that (i) DILIGENT supports the collaborative development of a shared ontology in a decentralized setting, (ii) the ontology needs to evolve, (iii) non-ontology engineering experts adapt their ontologies and participate in the ontology engineering process, and (iv) the organizational structure DILIGENT suggests fits the organizational setting found in the IBIT case study.

5.4.1.3. Organizational Background

The IBIT case study takes place in the tourism sector of the Balearic Islands. A number of organizations participating in the case study want to collaborate on some regional issues. Therefore, they collect and share information about *indicators* reflecting the impact of growing population and tourist fluxes in the islands, their environment and their infrastructures. Moreover, these indicators can be used to make predictions and help planning. For instance, organizations that require *Quality & Hospitality management* use the information to better plan, for example, their marketing campaigns. As another example,

a governmental agency, a Balearic government co-ordination center of telematics, provides the local industry with information about *new technologies* that can help the tourism industry to better perform their tasks.

Due to the different working areas and objectives of the collaborating organizations, it proved impossible to set up a centralized knowledge management system or even a completely centralized ontology. The case study partners asked explicitly for a system without a central server, where knowledge sharing is integrated into the normal work, but where very different kinds of information could be easily shared with others.

5.4.1.4. Case Study Description

We now describe the initial building stage, and the two rounds following the DILIGENT process.

a) Central Building

Roles Two domain experts with the help of two knowledge/ontology engineers built the first version of the shared ontology. In this case, domain experts were also knowledge providers and users.

Input The ontology engineering process was embedded in the application design development. According to the use cases the application should connect the employees of the seven organizations and provide access to databases. The use cases described the functions and roles of the users in their organizations, their work environment and technical infrastructure. The use cases elaborated on the interactions between the organizations and the exchanged knowledge. The relevant knowledge was grouped into the areas "sustainable development indicators", "new technologies" and "quality&hospitality management". For the area of sustainable development indicators detailed guidelines were available.

Activities The ontology engineering (OE) process started by identifying the main concepts of the ontology through the analysis of competency questions and their answers(1)[4]. The most frequent queries and answers exchanged by users were analyzed(2). The main objective of the ontology was to categorize documents. The concepts identified were divided into three main modules: "Sustainable Development Indicators (SDI)", "New Technologies (NT)" and "Quality&Hospitality Management (QHM)". From the competency questions the board quickly derived a first ontology with 20 concepts and 7 relations for the "SDI" ontology. For "NT" the board identified 15 concepts and 8 relations and for "QHM" 8 concepts and 5 relations. Between the modules 8 cross module relations were

[4]The numbering here and in the following, corresponds to the number of the activity in Figure 4.2.

introduced. A part of the result of the initial building stage is visualized in Figure 5.1(a), page 122.

The first round of our OE process started with the distribution of the three modules of the common ontology to all users. In both rounds, users - during the local adaptation stage - and the board - in the central revision stage - could perform ontology change operations. They could *introduce* concepts/relations/instances, *delete* concepts/relations/instances, or combine these operations arbitrarily, thus *extend* or *restructure* the ontology. Most frequently the concept hierarchy was changed.

Tool support The conceptual model of the first ontology was designed on paper. We used OntoEdit to formalize and implement the shared ontology. At this point the standard functionality of OntoEdit was sufficient to build the first version of the shared ontology.

I.) First Round

The first month of the case study, corresponded to the first round of the DILIGENT process. One organization with seven peers participated. The local adaptation stage lasted for one month, while the central analysis and revision stage took two days.

Ib.) Local Adaptation

Roles In the first round of our process one user with OE experience and six users without OE background participated.

Input The first version of the shared ontology was distributed with the system. The users had access to their local information, *viz.* folders, documents, emails and bookmarks, and used OntoScrape to create RDF(S) representations from it.

Activities

Local analysis of the shared ontology (3) The users initially regarded the shared ontology mainly as a classification hierarchy for documents. Consequently they compared their existing folder structures with the shared ontology to *understand the shared ontology* and *identify commonalities between their own and the shared conceptualization*.

Local specification of new requirements (5) *Identification of missing conceptualizations* was thus based on mismatches between the shared ontology and their local folder structures.

Local modification of local ontology (6a) Users *changed their conceptualization* and introduced new concepts. Rather than manually changing their local ontologies they preferred to reuse their folder structures.

Local integration of reused ontologies to the local ontology (6b) The users imported their local folder structures and reused them to extend the shared ontology. As the documents stored in a folder are automatically aligned with the corresponding concepts, the integration entailed the population of the ontology.

Ontology use (4) With the alignment of folders and concepts the users *organized their knowledge according to that conceptualization*.

Argument provision (8) The board received several requests to modify the shared ontology by email. As the argumentation framework was still under development at this time, they did not adhere to any structure for their requests.

Tool support The users worked with the XAROP application. The users imported document and folder representations with OntoScrape. In the first round the users had to change the local ontology with the OntoEdit plug-in. They aligned folders and concepts and created new concepts manually. Remote local ontologies were accessible using the OntoEdit plug-in. While the functionality of XAROP was quickly understood, the adaptation of the ontology with an ontology engineering environment (OEE) proved to be very difficult for some users. Therefore, we decided to include some basic adaptation functions into the XAROP user interface.

Decisions The users modeled concepts in the shared ontology representing the topics of their core working area. They continued to modify their local ontology until it covered all relevant topics of their domain. They shared only information which they thought being interesting for the group. In the interviews they commented, that they would share more files at a later stage, when they would feel more confident with the system.

Output The first round of the process resulted in 7 adapted ontologies.

Ic.) Central Analysis

Roles The board consisted of two ontology engineers and two domain experts/users.

Input The local adaptations from seven users were collected. Additionally the board had access to the folder structures of those users.

(a) First Version of the Shared Ontology

(b) Second Version of the Shared Ontology

(c) Third Version of the Shared Ontology

Figure 5.1.: IBIT Case Study: Evolution of the Shared Ontology (QHM)

Activities

Information collection from users (9): In our case we *gathered the locally updated ontologies and corresponding arguments*. In the first round the board (i) directly accessed the formal local changes on the different peers and (ii) some change requests on the conceptual level. At this stage the board also used (iii) the folder structures as indication for the requirements on the ontology, and it used (iv) the number of documents related to the concepts of the ontology as an indicator for its usage. In average a user shared 14 folders. Additionally, the board received new background knowledge which led to many additions in the "NT" module. The "SDI" module was

changed based on the formal changes collected electronically. Although the number of changes varied between the different modules the kind of changes was the same. Therefore, we subsequently focus on the changes introduced to the "QHM" module which are partly visualized in Figure 5.1.

Analysis of the obtained information (10): The board *analyzed the changes introduced* by the users at a conceptual level. They can be categorized as follows:

Elaboration The elaboration of the ontology was the most often observed action. The board could identify elaborations in three different ways. (i) The users correctly requested either formally or informally to add sub concepts to existing concepts to specialize them. (ii) The users incorrectly added new top level concepts, which were specializations of existing concepts. (iii) Finally they incorrectly refined the wrong concepts. In this way users elaborated the "NT" module with 15 concepts, the "SDI" module with 3 concepts and the "QHM" module also with 3 concepts.

Extension The board regarded a change as an extension whenever users requested new concepts on the top level. Again, users could not distinguish whether a required concept was an elaboration or an extension. Users extended the "NT" module with 2 concepts and the "QHM" module also with 2 concepts. The "SDI" module was not extended.

Renaming In some cases the users liked the way the board had conceptualized the domain, but did not agree with the names of the concepts. This happened twice in all modules.

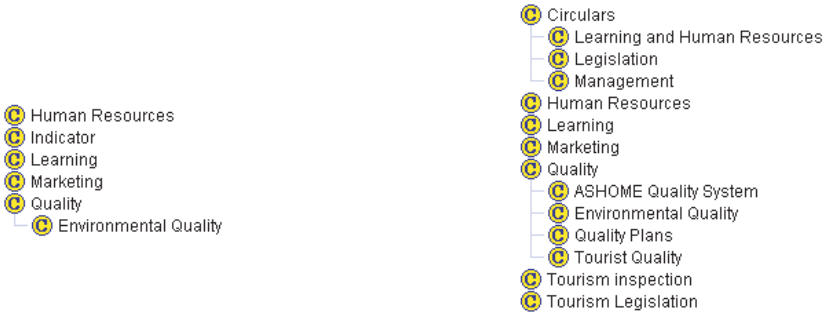
Usage Usage behavior of single concepts in the common ontology was analyzed. This included (i) the number of queries posed to the system containing a specific concept, (ii) the number of documents related to that concept and (iii) the elaboration of a concept. Most of the users did not delete any concepts or asked to remove concepts. Nevertheless, the board concluded that a concept which was never used should be removed.

Specification of new requirements (12) The board *identified the changes presumably relevant for a significant share of all participants*. Indeed they decided to introduce all change requests into the common ontology since all were supported by at least two users either through usage or extension/elaborations. Moreover, the domain expert could provide reasonable arguments for the introduction of all changes. Thus, the division of the ontology into 3 modules already generated a consensual group of users.

Tool support The board collected the remote local ontologies with the OntoEdit plug-in and they also received some local ontologies by email. They used the clustering mechanisms to structure the analysis of the local ontologies. The board was satisfied with the functions offered by the plug-in.

Decisions All changes were motivated by user requests and changes.

Output: The analysis of the local adaptations resulted in 27 changes for the "NT" module, 10 changes for the "QHM" module, and 5 for the "SDI" module.



(a) User Extension to the First Version of the Shared Ontology

(b) User extension to the Second Version of the Shared Ontology

Figure 5.2.: User Extensions to the Shared Ontology

Id.) Central Revision

Roles Two ontology engineers and one domain expert participated in the central revision stage.

Input In the analysis stage the board collected on total 41 changes to all modules. For example, 8 change requests related to the "QHM" module. These changes were formalized in this stage in order to update the shared ontology.

Activities

Customization of shared ontology (14) The new requirements and conceptual change requests identified in the analysis stage were translated into the following modeling decisions.

Concept elaboration For the "QHM" module over 50% of the changes were elaboration. In this case all elaborations were also new concepts. For example two users extended the concept Quality with the concept Environmental Quality. The requirement of a more detailed conceptualization was the main reason provided by the users for concept elaboration.

Concept removal Concept removal is less often observed. Only one concept was removed, because it was not used by any of the participants.

Concept renaming Some concepts were renamed, because synonyms were more common than the originally chosen term. The old name was kept for documentation purposes.

Concept creation In the "QHM" module the board introduced 2 new top level concepts, namely Learning and Tourism Legislation.

Relation creation In addition to the new and changed concepts the ontology engineers created 3 new relations.

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. The folder name "SustInd2002" matches the concepts Sustainable Indicator and Year[5]. 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. A concept had been used, if a user created instances of it, aligned documents with it, included it in a query or created 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.

Evaluation of new shared ontology (15) The board compared the requirements with the new version of the shared ontology and ensured that all requirements were met. Furthermore, the board checked the naming conventions.

Documentation (16) The board recorded the number of users contributing local changes and the names of the board. In our case all decisions were unanimously

[5] Year is sub class of class Temporal

taken, however in case of conflict the decision would have been achieved by majority voting. The board decided to collect the local adaptations after 2 months.

Distribution of new shared ontology (18) The new shared ontology was sent to the participants by email. The board assisted users solving technical updating problems.

Tool support Standard functionality of OntoEdit was sufficient in the central revision stage.

Decisions It was decided to model the requested changes in a way that ontological and user requirements were met.

Output After modeling the conceptual changes, the second version of the common ontology contained 54 concepts and 13 relations (Figure 5.1(b)).

Ie.) Local Update

Roles The new shared ontology was distributed to the seven users of XAROP.

Input The users received the ontology serialized as an RDF(S) file.

Activites

Control of new shared ontology (19) This activity was not executed explicitly, but part of the next activity. The separation is a result of the case study.

Local analysis of changes in the new shared ontology (20) The users compared the new version of the shared ontology with their local ontologies. All users decided to update. They requested indicators to facilitate the analysis activity.

Integration of new and old version (21) The users *tagged their old ontology* by coping the local node repository file to a new location. The updated version of the shared ontology was *locally included* by overwriting the old version. This is a simplistic approach, but acceptable as the system was a prototype. As the updated version of the shared ontology was more detailed and covered more domain knowledge, the users could *align* more of their folders with common concepts.

Tool support The users aligned folders and the new shared ontology with the OntoEdit plug-in.

Decisions All users decided to use the new shared ontology as it covered more domain knowledge and they found their requests integrated to it.

Output As a result of this stage the new shared ontology was commonly used and the user folders were aligned with the new shared ontology.

II.) Second Round

In the second round the case study was extended to 4 organizations with 21 peers. The users participating in the first round had more experience. The local adaptation stage lasted for two month. The central analysis and revision stage took place at a two day face-to-face meeting.

IIb.) Local Adaptation

Roles In the second round of our process all seven users participating in the first round were still active. For the second round the case study was extended resulting in 14 more users from three additional organizations. None of the new users had OE experience.

Input The experienced users started with the result of the local update stage, while the new users received only the new shared ontology. All users shared the local information which they thought relevant for the groups.

Activities As in the first round participants changed and used the common ontology according to their needs. Due to the larger number of participants more modifications were introduced. In particular the module "QHM" evolved (*cf.* Figure 5.2(b)). Regarding the activities defined for this stage, the users repeated the tasks already described for the first round. As more information was available in the overall system, they told us that they had used it more extensively.

Tool support Besides the tools applied in the first round, we added to the XAROP interface the possibility to change the ontology without using OntoEdit. The users could also retrieve remote local ontologies directly within that interface. The users appreciated this possibility very much as they had not to use another tool. The possibility to automated the alignment process also introduced in the second round was not used.

Decisions The new users behaved similar to the users in the first stage and did not share many folders, as they wanted to gain confidence in the system first. The experienced users, however, published more information, and adapted the local ontologies accordingly.

Output The second local adaptation stage resulted in 14 adapted ontologies. The rest of the users did not make changes. Although some did not change the shared ontology directly, they submitted change requests to their supervisor, thus they delegated the task of modelling their domain. The supervisor then communicated the requests to the board.

IIc.) Central Analysis

Roles In the second round the board consisted of one domain expert and two ontology engineers. Additionally two users were invited to answer questions to clarify the changes they introduced.

Input The 21 local ontologies of the users were the input to the second round. Some of the users did not change the common ontology at all, though all of them populated it. In a very hierarchical and well defined organization one single ontology could be adopted by all peers.

Decisions The board had to perform reverse engineering on the formal local ontologies from users in order to get conceptual models from them.

Activities

Information collection from users (9) As in the first round the updated ontologies were retrieved electronically. Some of the modification requests were collected interviewing the participants.

Analysis of the obtained information (10) Similar to the first round the modifications did not follow good ontology building practices. With respect to the conceptual modeling decisions the board observed that this time the users modified the ontology on a deeper level than in the first round. Renaming was a bigger issue in this round due to political changes, which required the adoption of new naming conventions. Moreover, generalization took place in two cases. Users introduced concepts which were more abstract than existing ones. The board moved one concept "Indicator" to another module of the ontology, since there the users elaborated it extensively.

In Figure 5.2(b) we observe that a user has extended the local version of the common ontology with a concept for Circulars. With the help of the domain expert, and taking also into account other local updates, the knowledge engineers inferred on the conceptual level that the module lacked concepts for Business Administration. Hence, the board not only introduced new concepts but also a generalization to existing ones 5.1(c). To exemplify an argumentation thread in favor or against a modelling decision we have selected the local extension Circulars performed by one user. Legislation was introduced as a subclass of Circulars. The argumentation

for a different modelling was straightforward, because the board found a Counter Example in form of a document dealing with Legislation, which was not a Circular. The most convincing arguments were selected and emphasized for documentation purposes.

Specification of new requirements As in the first round the board included all change requests from users. Again, as in the first round, only few of the concepts in the common ontology were never used.

Tool support Tool support did not change in comparison with the first round.

Output The board identified 3 changes in the "NT" module, 28 modifications for the "QHM" module and 15 for the "SDI" module.

IId.) Central Revision

Roles Two ontology engineers and one domain expert participated in the second central revision stage.

Input The 46 change requests on the three modules were the input for the second central revision stage.

Decisions All conceptual requests could be modeled.

Activities

Customization of shared ontology (14) The extensions to Quality were identical to all users. In this case there was an implicit agreement on how to model this part of the domain.

In the case of "Business Administration" only after discussions with the domain expert could the board understand the intended meaning of the extension. Hence, taking that as a starting point the board looked for the most adequate modelling, trying to find a balance between abstraction and user needs. The board did not want to introduce too abstract concepts, but in the case of "Legislation" there was a need for a more abstract concept than just "Tourism Legislation"

Renaming was performed following the most often used names for concepts. In other cases the discussion with the domain expert revealed that a political change was the reason for it, hence the renaming was performed although not all participants had yet adhered to the new names.

Documentation As in the first round, the board documented and explained the intended meaning of the newly introduced concepts. The documentation was mainly done by the domain expert. He paid explicit attention to the fact that not all users were familiar with all concepts of the ontology. Regarding renaming, the old names of the concepts were kept in the documentation.

Tool support Tool support did not change in comparison with the first round.

Output The third version of the common ontology contained 95 concepts and 15 relations (Figure 5.1(c)).

IIe.) Local Update

Roles The updated shared ontology was sent to the 21 participants in four organizations.

Input The stage took as input the new version of the shared ontology.

Activities The case study ended after the distribution of the new shared ontology. We collected feedback from the user w.r.t. to their impression of the new version. They emphasized that the new version represented their requirements at that time. The users commented that they appreciated being involved in the development process, although they recognized that they were not experienced in ontology engineering. They did not object to the modeling decisions of the board and understood the reasons for the differences between their change requests and the final modeling.

5.4.1.5. Lessons Learned

The case study helped us to better understand the use of ontologies in a decentralized environment. The set-up phase for DILIGENT was fast and the users quickly deployed the XAROP system. The users did utilize the ontology mainly as a classification hierarchy for their documents. Hence, they created few instances of the defined concepts.

Regarding our objectives, we conclude that DILIGENT indeed supports the collaborative development of a shared ontology. DILIGENT guided different people from different organizations in the process of collaboratively update and agree on a shared ontology. Furthermore, we could observe that the ontology needed to evolve, for example due to political change. The involvement of the users was also beneficial. Although our users were not trained in knowledge engineering or ontology engineering they could adapt the local ontologies. In this way they could profit from their own proposals (local adaptations) immediately. The result was much closer to the user's own requirements and other

users profited from it. From an ontology engineering perspective the user changes rarely followed good modeling practises. The intentions, however, were always recognized and could be translated straightforwardly into requirements and eventually be implemented.

Another finding was that DILIGENT can be adapted both to hierarchical and to more loose organizations. In hierarchical organizations not all actors change the ontology. Some want to delegate the responsibility to update the ontology to their hierarchical superior according to the organizational needs. In more loose organizations each actor will have his own local adaptation to best serve his own needs. DILIGENT processes cover both traditional OE processes and more Semantic Web-oriented OE processes.
We found that folder structures can serve as a good input for an ontology engineer to derive requirements for a domain ontology. Regarding the documentation of the shared ontology it was helpful to list the participants in the original building process, so that it is clear who was involved in the initial design decisions. For later stages of the process it was helpful to record the user names participating in the ontology evolution process.

In the course of the case study we realized that the agreement process became more complex with a growing size of the shared ontology and the increased number of participants. The emails to request modifications of the shared ontology were also difficult to analyze as more requests were sent to the board.

In spite of the technical challenges, user feedback was very positive since (i) the developed tool was integrated into their daily work environment and (ii) provided beneficial support to perform their tasks.

5.4.2. The AIFB Case Study

The experiences in the IBIT case study and the requirements of our scenario lead to the definition of the argumentation framework and the hypothesis that a structured argumentation process can facilitate ontology engineering discussions. On the one hand the argumentation framework defines a process to provide arguments, on the other hand it proposes to use a restricted set of arguments to focus the discussion.

The AIFB case study was set up in order to validate the hypothesis that a structured argumentation process with a predefinition of preferred argument types focuses the argumentation process and makes it more efficient. The case study setup was preceded by a literature review. The evolution of the taxonomy of life is a well documented historic example for the creation of a concept hierarchy from which we could derive an initial set of preferable argument types.

5.4.2.1. Review of Taxonomy Building in the Biology Domain

The biology domain is well suited for a retrospective survey on ontology development processes. Although biologists do not call the taxonomy of living things an ontology, it still has many features which are relevant for ontologies, too. Since the taxonomy of living

things is essential for those studying, classifying and understanding life it is also very well documented. The available documentation is used to study the line of reasoning applied to introduce changes to the taxonomy of life. This review uses RST to classify the argument types used in the line of reasoning. Subsequently the AIFB case study reveals, whether the argument types used in the biology domain correspond to the ones preferably used in ontology engineering discussions. The analysis may thus strengthen the findings.

The taxonomy of organisms has been evolving since 1735 for over 200 years. Lineaus setup the taxonomy in order to classify living things according to their features. The classification of organisms is a major activity in the biology domain. Biologists use the classification to communicate between each other and to recognize similarities between living forms in different parts of the world. There is now the tendency to arrange organisms in the "Tree of Life" according to their evolution. Organisms which departed later in earth history should be closer in the tree than organisms which departed earlier. The possibilities of the molecularly sciences offer an ever more exact determination of the ancestors of a particular organism. Thus the Tree of Life has changed many times and will continue to do so, due to the discovery of new organisms and new analysis methods, such as DNA analysis.

The next paragraphs examine the development of that taxonomy and pay attention to the general development process and the agreement process.

Data Collection Process A number of journals exist where researchers submit proposals for changes to the taxonomy. We examined discussions on the National Center for Biotechnology Information (NCBI) Web site `http://www.ncbi.nlm.nih.gov/`.

Organizational background The evolution process of the Tree of Life since 1735 is comparable to the 5-stage DILIGENT process with many iterations. 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 organisms 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 organism in the taxonomy.[6] Given the

[6]Names are reused in different kingdoms.

difficulty and similarity of some names, the ever evolving new knowledge about ever growing number of organisms, and the difficulty of making available up-to-date knowledge to all stakeholders about so many organisms, several problems in designing and managing this complex and live/dynamic taxonomy arose (The Economist, 2006b). 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.

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 organisms 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.

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 accommodate new organisms, and once in a while **revise** the common understanding of the domain.

The major force for reorganization of the taxonomy over time has been the identification of important classifying features and gathering all organisms 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.

Currently, given the advances in molecular biology, the tendency is to use a cladistic approach, in which the taxonomy is organized according to the evolutionary relationships between life 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.

Arguments We have analyzed the publications, where scientists propose changes to the taxonomy of life. We have looked in detail at the structure of their publications and the way the support their change requests. They use arguments in favor or against certain taxonomic arrangements in order to make their points. We have then studied the arguments from a RST perspective.

When analyzing the arguments exchanged by taxonomists to change the names and organization of the taxonomy one can perceive its vast array and complexity. The following

example illustrates this complexity[7].

> ...*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
> ...

The analysis of the argumentation revealed that biologists, although using all kinds of arguments, use some kinds more frequently than others. They use arguments, such as examples/evidence, counter examples, elaboration, alternatives and comparisons to convey a certain decision.

Lessons Learned The examination of the Tree of Life building process is comparable to the DILIGENT process model. Researchers contributing to the Tree of Life evolution propose changes in designated journals. The growing number of discovered organisms makes the comparison between described species and new species increasingly difficult. RST proved to be a useful method to analyze the arguments exchanged in the journal. The examination of several change requests suggests that a few argument types play a major role.

The next sections present the AIFB case study which were setup to analyze the argument types exchanged in ontology engineering discussions. The review of the taxonomy of organisms evolution strengthens the hypothesis, that some argument types are more relevant in modeling discussions than others and that RST is an adequate technique to analyze such discussions.

5.4.2.2. Organizational Background of the AIFB Case Study

Based on the RST analysis of arguments that are exchanged and used to support changes in the taxonomy of organisms, we formulated as 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[8]. Arguments in collaborative, distributed settings take place in a social environment. Therefore, organizational issues are non negligible and were also taken into account.

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

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

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 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.

5.4.2.3. Objectives for the AIFB Case Study

For the AIFB case study we had three objectives. Our first objective was to confirm that RST is an adequate method to analyze ontology engineering discussions from an argumentation point of view. Our second objective was to identify the efficient argument types in such a discussion. Finally we wanted to define a process model for ontology engineering discussions.

5.4.2.4. Data Collection Process

First experiment The participants met in a virtual chat room. Each one using an individual 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 modeling 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.

Second experiment 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 where 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 discussed with the allowed argument types. 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 restrictive. Whenever needed, the moderator called for an example of an argument to enforce the participants to express their wishes clearly.

5.4.2.5. Case Study Description

First Experiment

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

Example [9] 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 Web site*, that people get also `informed about hobbies` Purpose
> **cs :** so we have person Restatement
> **jt :** what I find a bit more interesting is `the conference problem` Motivation
> ...

In the beginning participants brought forward different kinds of arguments, like background knowledge, examples, elaboration and so on. This led to different argumentation threads where 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, the 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 modeling 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, because timely moderating multiple threads is very difficult: by the time an intervention was issued two or three other interventions from participants had already

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

been issued. As a result, a core ontology with two concepts, Role and Topic, was agreed upon although more concepts were discussed.

We analyzed the discussion with the help of RST. Table 5.3 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 where participants shared a very similar background knowledge, the creation of a shared conceptualization without any guidance is 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.

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 5.3.: Argument Selection

Second Experiment

The goals of the second experiment were (i) to underline that with an appropriate argumentation framework the ontology creation proceeds faster and more effectively and (ii) the

resulting ontology represents a shared view.

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 management 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 special than km. If I manage something then 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
> ...

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 modeled 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.

5.4.2.6. Lessons Learned

The comparison of the two experiments shows, that the DILIGENT argumentation process guides ontology engineering discussions and makes the agreement process more efficient. The predefinition of argument types structures the discussion and makes it easier to follow. In some cases it was difficult support an argument with an evaluation due to the lack of appropriate evaluation measures for ontologies. In this case an agreement on evaluation criteria may be helpful.

The experiments also show that the analysis of ontology engineering discussions with rhetorical structure theory is possible. The analysis is difficult, though, if the discussion is tangled. The analysis revealed, that some argument types are better suited to focus the discussion and reach agreement than others. Restricting the allowed argument types in ontology engineering discussions streamlines the agreement process and makes it easier to follow. In a tool templates may be used to facilitate the provision of the restricted set of argument types. The most focusing arguments types were identified as elaboration, evaluation & justification, alternatives, example and counter example.

5.4.3. The Legal Case Study

The legal case study is part of the SEKT project, an integrated project funded by the European Union.[10] The main objective of the SEKT project is to integrate research in the fields of ontology management, human language technology and machine learning in order to facilitate knowledge management. The results of the project are applied in three case studies. The legal case study is concerned with application of knowledge management in the legal sector. The research results of the project shall be exploited by the companies participating in the project.

5.4.3.1. Objectives for the Case Study

As part of the SEKT project, the legal case study serves as a test bed for the technologies and methods developed within the SEKT project. From a methodological point of view we wanted to examine if the argumentation framework supports non-ontology engineers in building an ontology. In particular our objectives were: (1) to test the wiki tool to provide arguments and (2) the applicability of the restricted set of arguments.

[10] http://www.sekt-project.com

5.4.3.2. Data Collection Process

The data in the legal case study was collected using interview techniques. We interviewed the case study participants four times in 5 to 6 month intervals. We interviewed the same 4 to 8 people each time. Two of them organize the ontology building process and push the development, while the others participate irregularly. The interviews were complemented with tutorials in ontology engineering in general and the use of the argumentation framework taking into account the specific problems of the participants.
The first interview was held in the beginning of the project. At that time the domain experts had done ethnographic field work and collected competency questions, literal transcriptions of interviews, and completed questionnaires from the prospective users. The focus of the interviews was on the organizational setting of the case studies. In particular we wanted to learn more about the domain for the ontology, the kind of application to be developed, the usage scenarios, the potential users and the available information sources.
At the second interview the case study participants had started ontology building but could not agree on an ontology covering all requirements. Our main concern at that point was to determine the participants level of experience w.r.t. ontology engineering and their main problems with ontology building. In particular we asked them about their experience with ontology engineering methodologies, their knowledge of ontology representation languages and tools, and the process they followed to build the ontology.
At the time of the third interview a first version of the ontology had been build using the argumentation framework. Hence, we concentrated our questions on their experiences with the argumentation framework. We obtained detailed descriptions of the organization of the ontology building meetings, their frequency, the use of arguments, the wiki tool, the agreement process and the way of dealing with issues and ideas.
We interviewed the participants the forth time when the first *domestic violence* module of the ontology was finished and it had been tested in the target application. We identified the benefits of the argumentation framework and its problems for the legal case study. At that time ten month lay between the initial use of the argumentation framework and the last refinements of the ontology thus we asked them if they profited from the documentation of the argumentation.

5.4.3.3. Organizational Background

The goal of the legal case study is to provide support to professional judges. In the Spanish system one particular problem young judges face is when they are on duty and are confronted with situations in which they are not sure what to do (*e.g.*, in a voluntary confession, which process of questioning should be applied?). In such cases, they usually phone their former training tutor (experienced judges) for resolving the issue, but this is a slow procedure. In this case study, it is planned to develop an intelligent system to speed up the process and to relieve experienced judges from this effort by providing initial support to young judges. The *Iuriservice* prototype II will provide the judges with access to

frequently asked questions (FAQ) through a natural language interface. The system (iFAQ) should answer to the question posed by the judge with a list of question-answer pairs that offer solutions to the problem and a set of related and relevant case rulings. Thus, the software will be capable of clearing up doubts concerning judicial practice and caseload resolution by providing justified and uniform answers to the questions raised by newly recruited judges. Ontologies are being used to provide a more accurate search than a basic keyword search could offer. The accuracy and the validity of the knowledge repository is crucial. Only in the case that the requested knowledge is not in the system and cannot be reformulated from already stored knowledge, an experienced judge will be contacted. The result of this "expensive" consultation will be fed back into the system automatically.

Two national surveys have been conducted as a primary source of data regarding both the context of use and the contents of the questions to which the system shall respond. These surveys offer interesting and important data to elaborate the user profile. There are three aspects of the professional profile of judges most relevant to the legal case study. The first one involves the frequency with which new judges talk about the cases they are dealing with. Only 4.71% of the judges interviewed stated that they never exchange information concerning their cases with others, usually peers. Secondly, judges offer an interesting answer to the question of "which would you like to find if judges were given a Web service system?". The majority of them proposed a site where doubts regarding professional cases could be discussed. Finally, the surveys allowed us to identify questions related to three main areas which presented some difficulties to new judges: (i) the organization and management of judicial staff (clerks working in judicial units); (ii) the interpretation and implementation of new procedural statutes; (iii) the "on-duty" period.

The first version of the iFAQ system will cover questions from the "on-duty" period and will focus on issues regarding domestic violence. During the surveys nearly 800 questions from the "on-duty" period were collected form young judges. 200 of these questions are related to domestic violence. In the case study an ontology for this domain was developed, which is referred to as Ontology of Judicial Professional Knowledge (OJPK).

At the beginning of the case study the participants had no experience with ontology building or OEEs. However, they had studied existing ontologies for the legal domain and were aware of ontology engineering methodologies (Benjamins et al., 2005a). They concluded, that available legal ontologies are representation ontologies that support legal reasoning while the ontology required for iFAQ should model practical legal knowledge. In order to build the ontology for practical legal knowledge they selected METHONTOLOGY as the ontology engineering methodology to follow.

5.4.3.4. Case Study Description

The case study description is divided into two parts. In the first part we describe the experiences of the participants building an ontology without applying the argumentation framework while in the second part the participants adhered to it.

Starting Ontology Building

The domain experts had collected about 800 competency questions in ethnographic field work. In a first attempt to build the ontology for practical legal knowledge they wanted to extract all relevant terms from the competency questions in order to build a concept list. The concepts should have been a starting point for hierarchy building and the introduction of relations. They envisioned to integrate the built ontology with an existing legal core ontology aligning core legal concepts with concepts representing practical legal knowledge. In the following we report the experiences from this case study phase separating the specification activity from the conceptualization activity.

Roles In the first meetings four to eight persons participated in the ontology building process. All of them are trained in the juridical field, one is professor for legal philosophy. They all made their first experiences with ontology building during this case study, although they had studied methodologies to build ontologies for legal reasoning (Visser et al., 1997).

Input From the ethnographic study 800 competency questions had been collected. Furthermore a number of existing ontologies for organizing and structuring legal knowledge as well as for reasoning and problem solving had been analyzed (*cf.* (Valente, 2005)) w.r.t. the requirements for the Iuriservice II and in order to get a general idea of existing ontologies. No existing ontology provides a model for professional legal knowledge as it would be required for Iuriservice II.

Specification The specification of requirements for the OJPK started with the examination of existing legal ontologies.

As concrete law differs from country to country the first legal ontologies tried to conceptualize the general legal theory lying behind the different interpretations (Breuker & Winkels, 2003). In order to build an ontology for a specific application, an intermediate ontology or legal-core ontology links an upper-top ontology with the domain-specific ontologies.

With the intermediate level one decides the nature of law, takes into account the representation language for the legal knowledge and represents the legal structure. Existing legal ontologies, however, were built to support legal reasoning rather than legal processes.

The kind of legal knowledge relevant to support legal processes is not encoded directly in the law but it has to do with personal behavior, practical rules, corporate beliefs, effect reckoning and perspective on similar cases, which remain implicit and tacit within the relation among judges, prosecutors, attorneys and lawyers. This is a different kind of legal knowledge than the one modeled in Legal-Core ontologies and it is called professional legal knowledge (PLK) (Benjamins et al., 2005b).

Therefore, the OJPK requires not only to represent the legal, normative language of written documents (decisions, judgments, rulings, partitions, etc.), but also those chunks of professional knowledge from the daily practice at courts. One of the main features of PLK is that it is context sensitive, anchored in courses of action or practical ways of behaving. In this sense, it implies: (i) the ability to discriminate among related but different situations (*e.g.*, when is it really needed or required to issue an injunction of protection to prevent a woman being injured or murdered by her husband?); (ii) the practical attitude or disposition to rule, judge or make a decision; (iii) the ability to relate new and past experiences of cases (*e.g.*, the need to ground each new ruling on past jurisprudential decisions (legitimacy process)); (iv) the ability to share and discuss these experiences with the peer group.

An ontology of professional legal knowledge would meet this requirements for all legal professionals (judges, lawyers, law enforcement authorities, etc.). In our particular case we have before us a particular subset of professional legal knowledge belonging specifically to the judicial field. Therefore, we will term the conceptual specification of knowledge contained in our empirical data OJPK.

In conclusion the participants required a system that contains knowledge to answer questions about judicial professional knowledge and to link the answers with judgements, rulings, laws, etc. It was decided to separate the two areas and model them in two different ontologies, which should later be linked through mappings. The ontology modeling written documents has not been build yet, thus we refer in the following only to the OJPK module.

Conceptualization The conceptualization of the OJPK module started with the extraction of terms from the 200 competency questions related to the 'on-duty' period and domestic violence in particular. More than 200 concepts could be extracted. The compilation of that list was followed by several meeting each lasting for a few hours. In these meeting, which took place twice a month, the participants tried to build concept hierarchies and relate the concepts from the list. A middle-out strategy was followed generalizing and refining the selected concepts if necessary. The participants started discussions in order to agree on a shared conceptual model. After a good start the conceptualization did not proceed since the discussions took very long and were tangled, due to the following problems.

- The generalization of the identified concepts resulted in the introduction of concepts typically found in Legal-core ontologies. It was tried to build a coherent model for legal reasoning as well as for professional legal knowledge. This resulted in numerous discussions. Legal ontologies for legal reasoning make use of rules to derive a judgement from facts. In contrast the OJPK ontology models how the facts can be obtained if procedures can not be followed as foreseen in the textbooks. For instance, in one case there was no police to secure the scene of the crime so the judge had to do it, but did not know how.

- Although the participants reached agreement on some issues the same issues reoccurred at follow-up meetings. As they did not capture the results of the discussions in an appropriate way, the lines-of reasoning could not be recovered, and they resumed the discussion.
- The agreement process was time-consuming, as each legal concept has a long history, and the participants tended to consider all aspects of it before achieving consensus. The discussion took place independent of its relevance for modeling decisions.

The case study partners were not satisfied with their progress and asked for support. In response to their request, they were introduced to the argumentation framework. The experiences applying the argumentation framework are reported in the next section.

Ontology Building within the Argumentation Framework

The second attempt to build the OJPK was organized according to the process proposed in our argumentation framework. *Roles* and *Input* did not change, except that the persons involved in the previous process had gained some experience. Ontology building meetings took place twice a month and lasted for one day each.

Activities The activity *Choose moderator* was not followed as suggested as none of the participants explicitly took the role of the moderator, because they all wanted to contribute to the discussion all the time. In the last interview, however, they acknowledged that they should have selected one, as it could have further improved the organization of the meetings.

Although it was tried to reach agreement for all conceptualizations the professor had the final saying in case of conflicting proposals (*cf. Choose decision procedure*).

The competency questions related to the 'on-duty' period were *specified as issues*. The issues related to specific problems in the 'on-duty' period where then grouped together. For example the issues related to domestic violence were separated from the ones related to car accidents. In order to *provide arguments and ideas* the legal experts team selected from the competency questions the nouns (usually concepts) and adjectives (usually properties). Once the terms had been identified, the team discussed the need to represent them within the ontology and their place within the taxonomy. As they followed the middle-out strategy, new issues were introduced to include generalizations or refinements of the identified terms if necessary. Finally, the relevant relations between those terms were also identified. As an example of the use of the middle-out strategy in the legal case study ontology and in relation to the competency questions analyzed above, modelers considered that the concepts auto [interlocutory decision], recurso [appeal], demanda [private/civil lawsuit] and querella [public/criminal lawsuit] needed to be represented in the ontology. Moreover, a concept documento [document] had to be created as all those terms auto, recurso,

demanda and querella describe different kinds of documents. The result was the construction of a more general concept from the specific terms found in the competency questions. However, the team also agreed that demanda, auto, recurso and querella were not only instances of documento but also constituted a specific class of documents used only within the judicial process. For that reason, documento_processal [procedural document] had to be created as a subconcept of documento. At the same time, there are different types of appeals and court orders stated in the questions, that have to be considered instances of recurso and auto. In this case, the terms where specified, not generalized.

The legal experts captured the exchanged arguments for the defined concepts. Documento was introduced because of these competency questions:

- ¿Cuál es el tratamiento de las **denuncias** manifiestamente inverosímiles o relativas a **hechos** que evidentemente carecen de tipicidad?[11]
- ¿Y si se trata de una **querella** que reúne todos los demás **presupuestos procesales** pero los hechos objeto de la misma carecen de relevancia penal o manifiestamente falsos?[12]
- ¿Ante quién debe interponerse el **recurso de reforma** contra la prisión, delante del juez de guardia o del juez que dictó el correspondiente *auto de prisión*? [13]

They provided justification for the concept

> "En las preguntas encontramos distintos tipos de documentos que deberían ser modelizados dado que son específicos del ámbito judicial."[14]

and an example

> "Denuncia, demanda, recurso y auto son documentos que deberían ser representados."[15]

Furthermore they elaborated on the concept and introduced as refinement documento_procesal

[11] The Spanish sentences were translated by one of the legal experts. *Translation:* What treatment get the obviously implausible **reports** or fact reports which clearly do not belong to the criminal domain?

[12] *Translation:* What happens when a **lawsuit** that fulfills all the **procedural requirements** but the facts of it have no criminal relevance or are obviously false?

[13] *Translation:* To whom the **appeal of reform** against imprisonment has to be filed, to the on-duty judge or to the judge who dictated the imprisonment court order?

[14] *Translation:* In these questions we find different kinds of documents that should be modeled since they are specific to the judges domain.

[15] *Translation:* Denuncia, demanda, recurso y auto are documents that should be represented.

> "No es suficiente con que estos documentos se consideren tipos de documentos sino que deberían ser una clase de documentos específica, ya que forman parte de un proceso."[16]

Besides summarizing the argumentation in the wiki they also proceeded more effectively, since they had ordered the competency questions according to the sub modules, such as legal documents, legal process, legal phase or roles. Hence, they conclusively discussed an issue at a time, proceeded with a similar next issue until all competency questions for the sub module were treated. If modeling decisions were unclear, they collected systematically the pros and cons for the idea and came to a decision. If the discussion related to an already discussed issue was resumed it could be terminated quickly either by adding new pros or cons to the existing line of reasoning or by finding that all pros and cons were already listed and did not need to be repeated.

They *decided on the issues and ideas* and built the ontology described in the next paragraph. To exemplify the importance of tracking the argumentation and the decisions we summarize the discussion regarding the role concept in the following:

The need for the role concept within the legal domain had also been contemplated in other relevant legal ontologies. In the legal core ontology (*cf.* (Breuker & Winkels, 2003)) role is a subclass of mental_entity, described as a functional view on a physical_object, agent_behaviour or mental_process. For these authors, roles are played by persons who are agents (Breuker & Winkels, 2003).

Another approach to model role is the one presented by Gangemi et al. (2003a) as part of the Jur-(Ital)Wordnet project, an extension to the legal domain of the Italian version of EuroWordnet. Jur-(Ital)Wordnet has been based on the DOLCE foundational ontology (Gangemi et al., 2003b). In the preliminary linking of legal concepts to DOLCE , Jur-(Ital)WordNet, contains that natural_person (considered a physical_object) is separated from functional roles. Under this point of view, judge, defendant and prosecutor would be functional roles, whether or not they are physical objects. The legal experts decided that one agente [agents] might play several roles during a process and might have several open processes where it plays different roles.

Output The OJPK has, currently, nearly 50 concepts, 100 relations and more than 300 instances (*cf.* Figure 5.3). At the moment, some top classes of the domain ontology have been identified: acto_procesal, órgano_ judicial, calificación_jurídica, rol_procesal, documento_procesal, fase_procesal, jurisdicción, proceso_judicial, profesión_jurídica, and sanción.

Acto_procesal [procedural act] represents a specific action taking place in the course of a judicial procedure. A subclass of acto_ procesal is acto_de_comunicación

[16] *Translation:* It is not enough to have these as kinds of documents, but a specific class of documents since they are part of a process.

[communication act], a class that includes all those acts of communication made by the court.

Órgano_judicial [court] is a subclass of agente [agent]. It is a class of organización [organization] and can perform actions with or without consciousness.

Persona [person] is also a subclass of agent.

Calificación_jurídica [legal status] is a required class which consists of all those types of crimes, felonies, misdemeanors or legal status regulated by norms or established by final rulings.

Fase_procesal [procedural phase] is an important concept for the OJPK ontology as it represents the time phases in relation to the judicial process. This concept is subclass of fase [phase].

Proceso_judicial [judicial process] is a key concept for the OJPK ontology, as most of the questions are somehow related to procedural problems during on-duty periods or during normal opening hours.

Rol_procesal [procedural role] is a subclass of role. A role is the part that an agent plays in a specific situation.

Documento_jurídico [procedural document] is a subclass of documento [document].

Jurisdicción [jurisdiction] and sanción [sanction] are relevant concepts regarding the geographical distribution of courts and the different types of sanctions (derived from civil or criminal liability), respectively.

Some properties/attributes of concepts and relations between concepts have also been identified and some are summarized in the following list:

Agente
- has_role {instances of rol}
- is_involved_in {instances of hecho [event]}
- has_state {instances of estado [status]}
- has_location {instances of localización [location]}

Acto_procesal
- has_document {instances of documento_procesal}

Fase_procesal
- begins_with
- ends_with
- followed_by {instances of fase_procesal}
- has_time_interval

Proceso_judicial
- has_phase {instances of Fase_procesal}

Rol_procesal
- played_by {instances of agente & instances of profesión_ jurídica}
- has_time_interval

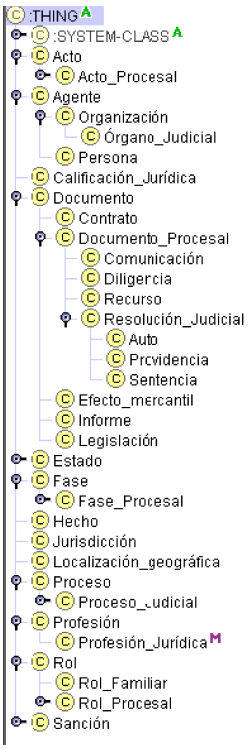


Figure 5.3.: Ontology of Judicial Professional Knowledge

Tools Ontology building was supported by several tools used in parallel[17]. These tools were the wiki, used for the tracking of the argumentation, the KAON OI Modeler, used for the visualization of the ontology, and Protégé, used for the formalization of the ontology. In Figure 5.4 we see a screen shot of the wiki, running on the SEKT portal, showing the concept Hecho with its description, argumentation and attributes, as well as a graph made with the KAON OIModeller showing the concept and its attributes.

5.4.3.5. Lessons Learned

Starting from our objectives the legal case study gives evidence that (1) the argumentation framework enables non-ontology engineers to effectively build an ontology, that it (2)

[17]The prototypical implementation of the tools lead often to work done more than once, due to the lack of interoperability of them

structures the argumentation process and that it (3) enhances traceability of design decisions. The wiki interface to capture the argumentation was sufficient to summarize the argumentation for each issue. It was not possible, though, to capture all arguments. For summarization the classification according to the five argument types *Elaboration*, *Justification*, *Example*, *Alternative* and *Counter Example* was straightforward. The summaries were helpful as the discussion proceeded and previous argumentations had to be recalled.

For the tools supporting the argumentation framework we were confronted with a number of deficiencies and strengths of using a standard wiki tool in combination with ontology editors. The partners of the legal case study appreciated the intuitive user interface of the wiki while they had problems with the ontology editors. Furthermore, they did not want to use more than one tool to perform the task.

The experiences with this setup lead to the following requirements for a future tool for supporting the argumentation framework:

- More push-technolgies need to be applied. Monitoring changes and the discussions of the ontology should be allowed: now, with the wiki, the user must actively look for changes in his domain of interest, but he cannot ask the system to actively tell him when a change occurs, *e.g.*, by email notification.

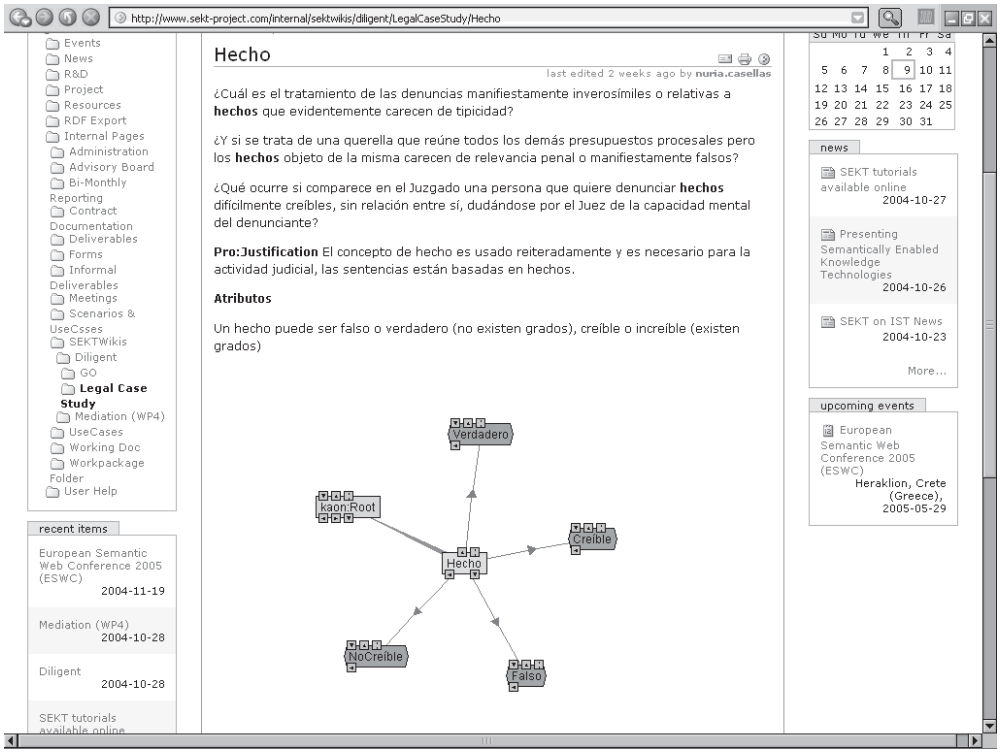


Figure 5.4.: Wiki-based Argumentation in the Legal Case Study

- A stronger integration with an ontology engineering environment will become crucial. For now the user had to keep the wiki up to date as well as the formalized ontology. In the wiki no semantic relationships exist between the different pages although the ontology would provide them.
- Visualization of the ontology is important. The users invested time to provide a visualization manually, even if it meant a lot of manual work. A future tool should include some kind of visualization and connect this to the captured argumentation.
- The classification of the arguments in the wiki was only text based. The advantages of the *Argumentation Ontology* did thus not materialize. In a specific tool the argumentation should not only be visually traceable but also semantically.

Overall the case study met our expectations and underlined the use of the argumentation framework.

5.5. Summary and Outlook

The case studies demonstrate that DILIGENT supports the building and maintenance of shared ontologies in decentralized organizational environments. Users like to agree on a shared ontology as it externalizes their shared interests. With the appropriate tools and an integration in the work environment ontology building can be performed by users. In this case some users change the shared ontology in order to adapt it to new requirements. Ontology evolution is important in order to keep the ontology in line with changing user requirements. The involvement of the users in the ontology building process was experienced as motivating.

Capturing the argumentation in ontology engineering discussions according to a predefined set of argument types structures the building process and makes the discussion more coherent. The legal case study shows, that the predefinition of argument types provides more fine-grained support for non-expert users. The semi-formal representation of arguments facilitates their later retrieval and allows for the detection of inconsistencies in the user argumentation.

We envision several directions for future extensions of DILIGENT. In particular for the application of DILIGENT in a business context the evaluation of the controlling activities in large scale ontology development processes is interesting. Furthermore, effort estimation incorporating accurate cost benefit analysis for ontology based applications is still an open issue. Effort estimation may also guide decisions related to the initial size of the shared ontology and its update cycle. Tool development is another aspect for future development. The development of plug-ins for other OEE than OntoEdit may increase the acceptance of the methodology.

Part III.

The REMINDIN' Routing Algorithm

"In order not to try all possibilities,
a resourceful machine must classify problem situations
into categories associated with the domains of effectiveness
of the machine's different methods."
— Marvin Minsky,
STEPS TOWARD ARTIFICIAL INTELLIGENCE

6. Routing in Semantic Peer-to-Peer Systems with REMINDIN'

Overview

DILIGENT is a methodology guiding users and ontology engineers in the process of building and evolving an ontology in distributed settings. REMINDIN' is a routing algorithm building on such ontologies to find requested information within such systems. Based on information about remote peers knowledge REMINDIN' selects a number of locally known receivers for a request and forwards it to them. REMINDIN' defines several strategies based on a shared ontology in order to ensure that (i) the right information can be found, (ii) the network is not flooded with messages, (iii) the local peer needs to maintain only a limited index. In this chapter we define the general problem of finding information in a distributed system. We then describe existing strategies to the problem at hand and present the idea behind REMINDIN'. The idea has been translated into a number of algorithms which are introduced afterwards. Chapter 7 focuses on the evaluation setup for REMINDIN' and its result.

References: This chapter is based on the publications (Tempich et al., 2004b), (Haase et al., 2004a), (Löser & Tempich, 2005), (Löser et al., 2005b), (Löser et al., 2005a) and (Tempich et al., 2005b)

6.1. Feasibility Study

In this section we motivate the design decisions for our routing algorithm. We derive the requirements for the routing algorithm from our use cases (*cf.* Section 3.1, page 33) in Section 6.1.2 and select an appropriate query routing model in Section 6.1.3.

6.1.1. Semantic Routing Use Cases

The development of the REMINDIN' algorithm was driven by the requirements from the two use cases of the SWAP project. The two use cases are representative for the general

distributed knowledge management (DKM) scenario. They have already been introduced in Section 3.1, page 33, thus, we emphasize the aspects relevant to routing here.

6.1.1.1. The IBIT Use Case

In the IBIT case study XAROP supports different organizations in sharing information about *sustainable development*, *new technologies* and *quality and hospitality management*. An ontology represents the relevant concepts for each of these domains (*cf.* Section 5.4.1). The participants use the ontology, *i.e.* construct complex queries, to search for information and they store relevant answers in their own knowledge base. The information they share changes often to account for the latest development for example in tourist arrivals and it may not be accessed by everybody. They want to search for information at remote peers which they know and get to know people sharing similar interests. They want to retrieve relevant information rather than all available information, because they are also prepared to call somebody if they do not get the answer quickly. As their infrastructure is not at the latest technological level they are short in storage and computing resources. Thus they do not run the P2P system all the time, but often leave and join the network. As some organizations are also competitors they have strict requirements w.r.t. to the accessibility to their local information. They may, however, give access to local information to remote peers depending on the request. For the same reason they do not want to publish their expertise in certain fields, as others could profit from this information. Besides it is a well know problem in knowledge management, that knowledgeable employees do not want to publish their expertise to prevent an overwhelming number of requests from other employees, while not so knowledgeable people overestimate their expertise in order to be promoted.

6.1.1.2. The Bibster Use Case

In the Bibster case study, we have explored the sharing of BibTEX information between peers of researchers. BibTEX is locally harvested from files and stored on each peer in the SWAP LNR. The BibTEX entries are assigned to topics defined in the ACM topic hierarchy either manually or automatically based on keyword matches of title strings and topics or both. Each researcher may search on the own peer as well as in the P2P network in order to retrieve the appropriate bibliographic data. This scenario is particularly interesting, because *(i)* BibTEX data have a stable interesting core, but also greatly varying additional fields as each user may define his own BibTEX entries; *(ii)* BibTEX data can never be fully captured in a centralized repository, because one repository, such as DBLP can only reflect a small set of topics (*e.g.*, databases and AI, but not organizational issues of knowledge management). The researchers search for BibTEX entries based on a combination of different properties defined in BibTEX and the ACM categorization of the entries. Researchers are interested in more than one result to a query, because answers might be complementary; they are not interested in all possible answers, though. Relevant result sets are integrated into the local repository. As in the IBIT case study researchers like to search

remote repositories of people they know and of researchers with similar research interests. For researchers the number of returned BibTEX entries is less relevant than their relevance to the request, because the time saved by finding complete BibTEX entries should not be consumed by the search process. For the same reason the network should respond quickly, even if the number of participants increases. Researchers often leave and join the network. As researchers need actual information on the latest research trends the entries they share change rapidly. In this scenario security issues are of no concern.

6.1.2. Requirements for Semantic Routing Algorithms

We have organized the requirements arising from our use cases according to the dimensions defined in Section 3.2, page 34.

Expressiveness The routing algorithm should support complex queries expressed in terms of the ontology. In both case studies we use SeRQL as a query language.

Comprehensiveness In both case studies the routing algorithm should support the retrieval of more than one answer to a search request.

Autonomy In the IBIT case study peers connect preferably to other peers in their organization but also to semantically close peers. They want to determine who can search and open their files. In the Bibster case study peers also connect preferably to colleges and researchers with similar interests. For researchers it could be interesting, who is interested in particular papers. Therefore, complete autonomy is required in both case studies.

Efficiency In the IBIT case study efficiency is relevant w.r.t. to resource consumption of the P2P application but due to the small number of peers not as relevant w.r.t. to network usage. In the Bibster case study network efficiency is relevant.

Quality of service Neither case studies requires that the routing algorithm supports the retrieval of all available answers to query.

Robustness In both case studies the network should cope with a high volatility w.r.t. the availability of peers and changing interests of the peers. We do not expect any denial of service attacks which might destruct the network. In neither case study a particular organization wants to take the responsibility to run and maintain the application for all participants.

Security In the IBIT case study access to information is restricted to peers explicitly authorized. Security is of no concern in the Bibster case study.

6.1.3. Selection of a Routing Approach

In order to select an appropriate approach to routing for our use cases we compare the requirements from our case studies with the advantages and disadvantages of the different routing paradigms (*cf.* Section 2.4.2.2, page 30).

The result of that comparison is summarized in Table 6.1 and further elaborated in the following.

Expressiveness In P2P architectures which build on the transmission of index information to different peers it is difficult to support conjunctive queries. In such networks keys or index terms from one peer are placed on different remote peers, thus finding information for a query with more than one query term requires a join of the information of different peers. In P2P architectures which have a centralized index this is not a problem.

Comprehensiveness In P2P systems building on purely decentralized architectures without structure it is not possible to guarantee that all answers to a query are found while for the remaining system architectures such a guarantee is possible.

Autonomy Only in purely decentralized architectures peers can connect to any remote peer, without being forced to establish an additional connection with a particular peer or a set of particular peers.

Efficiency Efficiency should be discussed according to two lines: network load and local resource consumption. Except purely decentralized architectures without struc-

[1] In cases that the requirements for the IBIT case study differ from the ones in the Bibster case study the first indicator refers to the IBIT case study and the second to the Bibster case study.

Requirement Category	**Purely Decentralized Architectures**		**Partially Centralized Architectures**	**Hybrid Decentralized Architectures**
	Unstructured	Structured Index	(Super Node)	(Client/Server)
Expressiveness	YES	NO	YES	YES
Comprehensiveness	No guarantee	YES	YES	YES
Autonomy	YES	NO	NO	NO
Efficiency	No guarantee	YES	YES	YES
Quality of service	No guarantee	YES	YES	YES
Robustness	YES	YES	YES	NO
Security	YES	NO/ YES[1]	NO/YES	NO/YES

Table 6.1.: Selection of a Routing Approach for Distributed Knowledge Management

ture the rest of the approaches to routing generate some network load when a peer goes online, as an update of the index information is transmitted to the indexing peers. As the information can be found at predictable places the maximal number of messages sent to answer a query is also predictable (Milojicic et al., 2002). In purely decentralized architectures no messages are generated when a peer joins the network, but there are no guarantees w.r.t. to the necessary number of messages to retrieve information.

Quality of service In purely decentralized architectures there is no guarantee that the requested information can be found even if it is available while the other approaches to routing guarantee it.

Robustness All architectures except the hybrid one are robust enough for our scenarios as we do not expect denial of service attacks.

Security Purely decentralized architectures using structured information to support query routing and thus have to distribute information to remote peers are not adequate for our scenario, as the local information may be distributed to any place in the network.[2] Therefore, only purely unstructured decentralized architectures fulfill the requirements of our scenario w.r.t. security. In the Bibster case transmission of index information to remote peers does not raise security concerns, while in the IBIT case that would be an issue.

Summary

From the comparison of the case study requirements with the properties of the different P2P architectures we conclude that only purely decentralized architectures without structure meet them w.r.t. autonomy and expressivity. The main challenge in these architectures, though, is to ensure sufficient efficiency for a requested quality of service. In the next sections we present the REMINDIN' routing algorithm developed for purely decentralized P2P architectures without structure, which meets the requirements just elaborated.

6.2. Foundations of the REMINDIN' Routing Algorithm

P2P networks based on a purely decentralized architecture without structured overlays are comparable to the social networks of persons where a peer corresponds to one person. Projecting the task of efficiently finding relevant information in a P2P network on such social networks we observe that people deploy successful strategies in everyday life to satisfy their information needs. Studies of social networks show that the challenge of finding relevant information may be reduced to the task of asking the 'right' persons. 'The right

[2] Corresponding to column *Structured Index*.

persons' are the ones who either have the desired piece of information and can directly provide the relevant content or the ones who can recommend 'the right persons'.

Milgram (1967) and Kleinberg (2000) experiments illustrated that people with only local knowledge of the network (*i.e.*, their immediate acquaintances) were quite successful at constructing acquaintance chains of short length by selecting 'the right persons'. The emerging networks have 'small-world' characteristics. In such a network, an information request is forwarded to those persons who are 'closest' to the destination. The request reaches the destination with a small number of hops although sender and receiver may be far away from each other, as some persons among the immediate acquaintances have far reaching links. We observe that such mechanisms in social networks work although

- people may not always be available to respond to requests,
- people may shift their interests and attention,
- people may not have exactly the 'right' knowledge, but only knowledge which is *semantically close*.

In the following we motivate the **Routing Enabled by Memorizing INformation about Distributed INformation (REMINDIN')** algorithm looking at the social interactions which enable persons to select among their acquaintances the appropriate ones.[3] We abstract from these interactions and introduce a number of overlay layers to model them. In this overlay layers each peer plays the role of a person in a social network.

6.2.1. Routing Based on Social Metaphors

We observe that a human who searches for answers to a question may exploit the following assumptions[4]:

1. A question is asked to the person who one assumes that he best answers the question.[5]
2. One perceives a person as knowledgeable in a certain domain if he/she knew answers to our previous questions.
3. A general assumption is that if a person is well informed about a specific domain, he/she will probably be well informed about a similar, *e.g.*, the next more general, topic, too.

[3] The algorithm was also presented under the acronym INGA (Interest Node Grouping Architecture). In this thesis we use the name REMINDIN', because the original idea was published under this acronym.

[4] We do not claim that these observations of social networks are in any way exhaustive or without exceptions.

[5] 'Best' in our current terms only means that he has the most knowledge. In future versions one may consider properties like latency, costs, etc.

4. A person who knows many others has a good chance to know somebody who can answer our question.

5. If someone (A) asks another person (B), B will remember A and his question. If the B has to answer the same question, he will then return the question to A — assuming that he has gathered valuable information in the meanwhile.

6. In some cases a person asks anybody either because no one else is available or because the other one is physically close.

REMINDIN' builds on the metaphors of P2P networks being like a human social network and adopts the above mentioned assumptions in an algorithmic manner.

6.2.2. Routing with Semantic Overlay Layers

In order to mirror the observations from social networks in our routing algorithm we introduce different semantic overlay layers to organize a semantic shortcut index. A shortcut in our system is a non forwarding indexing link for a specific item of the index. Each peer maintains its own personal semantic shortcut index and organizes the shortcuts according to four different layers. The shortcuts are used to select appropriate remote peers if the local peer sends a query or it forwards a query from another remote peer (*cf.* social metaphor 1).

- The best peers to query are those that already have answered the query or a semantically similar query successfully in the past. We call such peers *content providers* (*cf.* social metaphor 2).
- If no content providers are known, peers are queried that have *issued semantically similar queries* in the past. The assumption is that this peer has been successful in getting matching answers and now we can directly learn from him about suitable content providers. We call such peers *recommenders* (*cf.* social metaphor 5).
- If we do not know either of the above we query peers that have established a good social network to other persons over a variety of general domains. Such peers form a *bootstrapping network* (*cf.* social metaphor 4).
- If we fail to discover any of the above we fall back to the default layer of neighboring peers. To avoid overfitting to peers already known we occasionally select random peers for a query. We call this the *default network* (*cf.* social metaphor 6).

If a local peer needs to select remote peers it searches each of the layers for best fitting shortcuts. The basic principle laying behind the shortcut mechanism consists of dynamically adapting the topology of the P2P network so that the peers that share common interests spontaneously form well-connected semantic communities. Crespo & Garcia-Molina

(2002b) shows that each user is only interested in a rather limited number of different topics. Therefore, being part of a community that shares common interests is likely to increase search efficiency and success rate. We exploit the semantic relationships between the available shortcuts and the query in order to find most relevant ones (*cf.* social metaphor 3). If no relevant shortcuts can be found on one layer information from the next layer is applied. The value of selected shortcuts is evaluated each time we receive an answer from the network to one of our own queries. In the next section we provide more detailed information on the layers itself, their creation and maintenance.

6.3. The REMINDIN' Semantic Overlay Layers

REMINDIN' acquires knowledge about remote peers in the network at runtime of the underlying application. Depending on the kind of knowledge shortcuts are created on different layers: these contain an increasing amount of information about the specific knowledge of a remote peer. The **network layer** stores routing information about remote peers known from the physical network; the **bootstrapping layer** contains general information about the knowledge of remote peers; the **recommender layer** stores information about potentially knowledgable peers; and the **content provider layer** contains specific information about remote peers. Before we explain the layers in more detail, we introduce the knowledge and query model which constitute the basis of the layers.

Knowledge model REMINDIN' is an instance of the SWAP system peer selector component. As described before (*cf.* Section 3.3) knowledge in the SWAP system is stored as an RDF(S) model in the local node repository. We attach to each resource in the local node repository a Swabbi- and Peer-object. These objects contain the layer and rating information thus representing a shortcut. REMINDIN' can be applied to other knowledge models. It is important to note, however, that a shortcut *sc* is created in the form *sc(indexTerm, pid, queryhits, shortcutType, update)*, where *indexTerm* is the indexed resource, *pid* is a unique peer identifier, *queryhits* is the number of returned hits for the *indexTerm*, *shortcutType* is a reference to the layer and *update* refers to the time of the last update of the shortcut. In our specific case *indexTerm*s are resources of the RDF(S) model. In Section 6.4 we demonstrate the usage of the semantic information related to resources in order to select appropriate shortcuts.

Query model The query language used in the SWAP system is SeRQL. SeRQL supports conjunctive queries and other boolean operators similar to SQL. REMINDIN' requires that the queried resources can be extracted from the query.

Shortcut index size Shortcuts of any kind can be created to any peer in the network for any resource. However, due to limited local resources and each peer specific interests,

peers only maintain a bounded index of shortcuts. The decision of replacing a shortcut from the index, i.e. promoting new peers as shortcut acquaintances, depends on the history of the responses to previous requests issued by each peer.

Example The following example illustrates the definition of the various layers and the execution of the REMINDIN' algorithms. It considers a small P2P network consisting of three peers. The three peers share the ontology modeled in Table 6.2 and organize their knowledge accordingly. Table 6.3 summarizes the content stored in the LNR of the three peers. The example uses a small fraction of the ontology constructed for the IBIT case study.

Concepts	Relations
Document	*hasTopic* $\Longrightarrow$ Topic
Topic └ TourismActivity └ DestinationManagement └ TravelDistribution └ TourismTechnology └ BookingSystem └ GeographicalInformationSystem	

Table 6.2.: Example: Shared Ontology for Peers in the Network

6.3.1. Content Provider Layer

The design of the content provider shortcut overlay is comparable to existing work as published by Sripanidkulchai et al. (2003) or Cooper (2004) and helps to exploit the simple, yet powerful principle of interest-based locality.[6] That means if a content provider peer has a particular piece of content that another peer is interested in, it can be considered very likely that the content provider will also have other interesting items for that peer. A content provider shortcut is an indexing link to a remote peer which indicates that the remote peer has information about a specific indexing term.

6.3.1.1. Creation of the Content Provider Layer

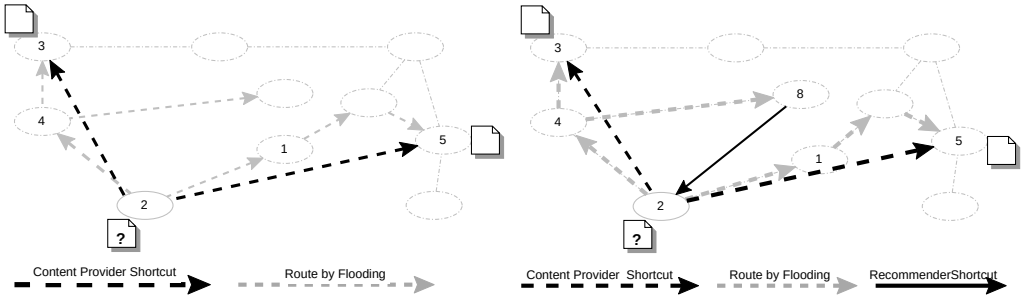
When a peer joins the system, it may not have any information about the interest of other peers. It first attempts to receive answers for its queries by exploiting less specific layers of the REMINDIN' peer network, such as the network layer, *e.g.*, by flooding. The lookup

[6] See the related work Section 8.3, page 217 for a detailed discussion.

returns a set of peers that store answers to the query. These peers are potential candidates to be added to the content provider shortcut list. Each time the querying peer receives an answer from a remote peer, content provider shortcuts to new remote peers are added to the list in the form: *sc(resource, pid, queryhits, 'c', update)*, where *resource* is the query terms taken from the query message, *pid* is the unique identifier of the answering peer, *queryhits* is the number of returned statements, *'c'* is the type of content provider shortcuts and *update* is the time, when the shortcut was created or the last time, when the shortcut was

Peer	Resource	No. of documents
2	TourismActivity	0
	└ DestinationManagement	0
	└ TravelDistribution	10
	TourismTechnology	0
	└ BookingSystem	0
	└ GeographicalInformationSystem	0
3	TourismActivity	10
	└ DestinationManagement	10
	└ TravelDistribution	10
	TourismTechnology	10
	└ BookingSystem	10
	└ GeographicalInformationSystem	10
	GeographicalInformationSystem ∧ DestinationManagement	5
5	TourismActivity	30
	└ DestinationManagement	50
	└ TravelDistribution	100
	DestinationManagement ∧ TravelDistribution	10
	TourismTechnology	0
	└ BookingSystem	0
	└ GeographicalInformationSystem	0
8	TourismActivity	0
	└ DestinationManagement	0
	└ TravelDistribution	0
	TourismTechnology	40
	└ BookingSystem	20
	└ GeographicalInformationSystem	100
	GeographicalInformationSystem ∧ BookingSystem	10

Table 6.3.: Example: Content Distribution for peers 3, 5 and 8



(a) Content provider shortcut creation (b) Recommender shortcut creation

Figure 6.1.: Resource Specific Shortcut Creation

updated. For conjunctive queries a content provider shortcut is created for each resource contained in the query.[7] The content provider shortcut list will grow with each submitted query until the maximum number of content provider shortcuts is reached.

6.3.1.2. Maintenance of the Content Provider Layer

Content provider and recommender shortcuts (*cf.* Section 6.3.2) are stored in the same list. As the number of created shortcuts increases the list fills up. If the maximum number of shortcuts stored in the index has been reached we delete existing shortcuts. Each time a short cut is added to the list, the total size of the index is calculated and if it exceeds the maximum index size the shortcuts are ranked according to their quality. The shortcuts of the lowest quality are deleted from the list. Quality criteria depend for example on the application domain. In Section 6.3.3 we define a number of quality criteria and propose a function to combine them.

Example Consider Figure 6.1(a). Peer 2 discovers shortcuts for the resource /Topic/-TourismActivity/TravelDistribution[8] by flooding the default network with a maximum number of hops (TTL) of three hops and creates two content provider shortcuts to peer 3 and peer 5. Another two content provider shortcuts are created for peer 1 and 4; as they have not answered the query the number of query hits is set to 0. The created shortcut index is listed in Table 6.4.[9]

[7] For conjunctive queries it is more difficult to update the number of statements for a particular shortcut, as only the number of statements for the combined query are known. Cooper (2004) proposed the use of moving averages to overcome this problem.

[8] In the examples the relation 'subclass-of' is indicated by '/' and the complete path is spelled out. The SeRQL query contains only the specific query resource in this case, *e.g.*, TravelDistribution.

[9] The calculation of the numbers in the different columns will be explained in the following sections. The *Maxsim* column indicates the similarity between the shortcut and the local content and is also explained later.

6.3.2. Recommender Layer

Very active peers issue many successful queries and produce many shortcuts. If a remote peer issues queries that are similar to the own interests, it can be beneficial to establish indexing links to this peer. The reason is that, if a remote peer has established a shortcut to an interesting content provider, it is likely that this peer will issue other queries on related resources that one will be interested in, too. Such recommender shortcuts thus represent a new kind of indexing links in the semantic overlay structure. If a peer can not directly determine a content provider peer for a given query, it can forward the query to the best matching recommender.

6.3.2.1. Creation of the Recommender Layer

In order to quickly fill up the shortcut index we create shortcuts for incoming queries that are routed through the local peer. Recommender shortcuts *sc(resource, pid, 1, 'r', update)* are created, where *resource* is the set of query terms from the query message. The *pid* for a respective shortcut is extracted from the query message as the PID of the querying peer. Since we will get no information about the number of results retrieved for the query, we set the number of query hits to 1. Finally *r* indicates the type of the shortcut for recommender shortcut and *update* is the time, when the shortcut was created or updated.

6.3.2.2. Maintenance of the Recommender Layer

As mentioned earlier content provider and recommender shortcuts are stored in the same index. If a new recommender shortcut is added to the list, we count the number of shortcuts and rank them according to their quality if the number exceeds the maximum index size. Shortcuts of the lowest quality are deleted from the list.

Example Consider again Figure 6.1(b). Peer 2 issues the query /Topic/TourismActivity/-TravelDistribution. Peer 8 creates a shortcut to peer 2 since this query was routed through

Pid	Resource	Query Hits	Maxsim	Type	Update	Rank
5	.../TourismActivity/TravelDistribution	100	1	c	45 sec	1
3	.../TourismActivity/TravelDistribution	10	1	c	40 sec	2
1	.../TourismActivity/TravelDistribution	0	1	c	19 sec	3
4	.../TourismActivity/TravelDistribution	0	1	c	20 sec	4

Table 6.4.: Example: Shortcut Index of Peer 2 after Query for /Topic/TourismActivity/TravelDistribution

peer 8. Table 6.5 represents the new shortcut index of peer 8. The same shortcuts with different similarity values are created at peer 1, 3, 4, 5 and other remote peers which receive the query.

Pid	Resource	Query Hits	Maxsim	Type	Update	Rank
8	.../TourismActivity/TravelDistribution	1	0,45	r	1 sec	1

Table 6.5.: Example: Shortcut Index of Peer 8 after query for /Topic/TourismActivity/TravelDistribution

6.3.3. Ranking Content Provider and Recommender Shortcuts

We assume that each peer can only store a limited amount of shortcuts, hence only knows a limited set of resource specific neighbors it can route a query to. If the local index size is reached a peer has to decide, which shortcut should be deleted from the index. For each shortcut in the index we compute a rank based on the following types of localities:

Semantic-locality We measure the maximum semantic similarity $maxsim$ between the resource of a shortcut and the resources represented by the local content of a peer according to Equation 6.3 on page 174. Hence, we retain a shortcut about resource t to a remote peer, if t is close to our own interests.

LRU-locality To adapt to changes in the content and interests we use a least-recently-used (LRU) replacement policy (Aho et al., 1971). Shortcuts that have been used recently receive a higher rank. Each local shortcut is marked with a time stamp when it was created. The time stamp will be updated, if the shortcut is successfully used by the local peer. There is thus an 'oldest' and 'latest' shortcut. The value $update \in [0..1]$ is calculated as difference between the shortcuts time stamp and the 'oldest' time stamp divided by the difference between the 'latest' and the 'oldest'.

Community-locality We measure how close a shortcut leads us to an answer. Content provider shortcuts, marked with a c, provide an one hop distance, we set $type = 1$. Recommender shortcuts, marked with a r require at least two hops to reach a peer with relevant answers, we set $type = 0.5$.

We weight the localities and compute the index relevance according to Equation 6.1.

$$\begin{aligned} relevance \quad = \quad & \frac{simInfluence * maxsim}{simInfluence + typeInfluence + timeInfluence} + \\ & \frac{typeInfluence * type}{simInfluence + typeInfluence + timeInfluence} + \\ & \frac{timeInfluence * update}{simInfluence + typeInfluence + timeInfluence} \end{aligned} \tag{6.1}$$

Shortcuts with the highest relevance are ranked at the top of the index, while shortcuts with a lower relevance are deleted from the index. Table 6.6 exemplifies the ranking of shortcuts.

Pid	Resource	Query Hits	Maxsim	Type	Update	Rank
5	.../TourismActivity/TravelDistribution	100	1	c	145 sec	1
3	.../TourismActivity/TravelDistribution	10	1	c	140 sec	2
3	.../TourismActivity/DestinationManagement	1	0,63	r	40 sec	3
8	.../TourismTechnology/BookingSystem	1	0,42	r	10 sec	4
1	.../TourismActivity/TravelDistribution	0	1	c	119 sec	5
4	.../TourismActivity/TravelDistribution	0	1	c	120 sec	6

Table 6.6.: Example: Shortcut Index of Peer 2 after a Number of Queries

6.3.4. Bootstrapping Layer

The content provider and recommender layer contain shortcuts to remote peers for specific resources. In cases a query can not be routed through these query specific shortcuts peer selection could be based on, *e.g.*, random selection of peers or on flooding. Adamic et al. (2001), however, shows that a high degree search strategy significantly reduces messages in unstructured networks in contrast to a flooding strategy. Bootstrapping shortcuts are links to peers that have established many shortcuts for different query topics to a lot of remote peers. We determine the bootstrapping capability by analyzing the in-degree and out-degree of a peer. We use the out-degree as a measure of how successful a peer discovers other peers by querying. To weight the out-degree, we measure the amount of distinct sources a peer receives queries from. We use the in-degree as a measure, that such a peer may share prestigious shortcuts with a high availability. By routing a query along bootstrapping shortcuts, we foster the probability to find a matching shortcut for a query and avoid the drawbacks of having to select peers randomly, *e.g.*, by flooding.

6.3.4.1. Creation of the Bootstrapping Layer

Each incoming query that is stored in our index includes the bootstrapping information of the querying peer. While a peer is online it continually updates its bootstrapping index based on incoming queries and stores bootstrapping shortcuts in the form *sc(pid, Bo)*, where *pid* is the PID of the querying peer and *Bo* its bootstrapping capability. Once an initial set of bootstrapping nodes is found, a peer may route its queries to the nodes with the highest Bo value. A peer calculates its Bo_pid value using Equation 6.2

$$Bo_{pid} = (1 + |outdegree|) \times (1 + |indegree|) \tag{6.2}$$

where *out-degree* is the number of distinct remote peers it knows. To compute an approximation of the *in-degree* without any central server we count the number of distinct peers that sent a query via the peer. In order to obtain a list of remote peers which know the local peer the local peer examines the message path of the queries routed through it. To avoid zero values we add one to both values.

In continuation of our example we assume that peer 2 has received the queries directly from peer 3 and 8 and no other peers have sent queries to peer 2. The Bo_2 value for peer 2 is than equal to $(1+4) \times (1+2) = 15$. The bootstrapping shortcut index contains then shortcuts for peer 3 and 8. We assume in our examples that $Bo_3 = 40$ and $Bo_8 = 25$.

6.3.4.2. Maintenance of the Bootstrapping Layer

We select bootstrapping peers only from the list of peers which are referenced in the shortcut index. There exist, thus, no separate maintenance mechanism for bootstrapping peers. As the selection of shortcuts is partly based on the similarity to the content of the local peer, not all peers in the network establish bootstrapping links to the same remote peers.

6.3.5. Default Network Layer

The default network layer facilitates the communication of the peers in the network. Different implementations for this layer exist. The SWAP system is based on the JXTA protocol. In JXTA peers register with a rendezvous server and publish their availability. Gnutella peers send ping messages to neighboring peers. Independently of the technique a peer establishes a number of connections to remote online peers to ensure that it can communicate at all.

6.4. Deploying Semantic Overlay Layers for Peer Selection

In the previous section we have introduced a number of shortcut layers to construct an overlay topology above the default network layer. Before we evaluate our routing approach and compare it with related routing approaches in the next chapter we describe the process and algorithms used to select appropriate remote peers from the different shortcut layers.

6.4.1. Peer Selection Process

REMINDIN' consists of several steps executed *locally* and *across the network* when *forwarding* as well as *answering queries* and when *receiving responses*. Assuming the user of a peer issues a query to the P2P network, the query is evaluated:

Locally against the local node repository. Its answers are presented.

Across the network: Recommending. Whenever a peer receives a query message, it first extracts meta-information about the querying peer and updates its bootstrapping and recommender index if needed. Then the REMINDIN' forwarding strategy is invoked to select a set of k peers that appear most promising to answer the query. The selection of the remote peers is itself a multistage process. The resources referenced in the predicates of the query are extracted. For each resource the shortcut index is searched and matching peers are selected (*cf.* algorithm 1). If no shortcuts exactly match the query resources the query resources are relaxed and semantically close shortcuts are searched for (*cf.* algorithms 5 and 7). This process is continued until either enough remote peers could be selected or a threshold is reached. The remote peers are ranked according to different measures. If no remote peers could be selected from the resource specific shortcut index know remote peers are ranked according to their bootstrapping capability (*cf.* algorithm 3). If the capability exceeds a certain threshold those peers are selected otherwise remote peers from the default layer are chosen (*cf.* algorithm 4). Finally the original query message is forwarded to k peers.

Across the network: Answering Queries. When a peer receives a query, it will try to answer the query with local content. We only return non-empty, exact results and route them directly to the querying peer. If the maximum number of hops is not yet reached, the query is forwarded to a set of peers selected as above.

Receiving Responses. On the arrival of result items a querying peer analyzes the message path and the respective number of results to create or update local content provider and recommender shortcuts. The sender of a query can also update its information about the remote peers he had originally selected for answering his query. If he was right about his assumption that the selected remote peers were knowledgeable about the query shortcut information can be updated accordingly. If the remote peers did not answer the query he can at least store that they did not know anything about the query and omit their selection the next time.

6.4.2. Peer Selection Algorithms

The REMINDIN' shortcut selection algorithm determines the candidate peers that are most promising to forward the given query to. The REMINDIN' strategy is based on the available local knowledge about the query resources as it is stored in the shortcut index of the peer:

- REMINDIN' only forwards a query via its *k best matching* shortcuts.
- REMINDIN' prefers content and recommender shortcuts over bootstrapping and default network shortcuts for forwarding queries.

- The REMINDIN' strategy constitutes a greedy *k best*-search heuristics. As such it may be led astray into a subnetwork of peers that appear to be the optimal choice from a local point of view, but that do not yield all the appropriate answers. To let the search escape such local optima, some queries are forwarded to a random set of peers.

  This randomness will later on show two major beneficial effects: First, it allows the individual peer to have a larger overview of the whole network and, hence, to establish the appropriate short distance *and* long distance shortcuts.[10] Second, it facilitates accommodation to volatility (especially in the form of new joining peers).

Algorithm 1 REMINDIN' Peer Selection:
peerSelection($O, Q, MP, k, t_{greedy}, rand, \mathcal{SC}, maxTTL, queryRelaxation$)

Require: **LocalNodeRepository** O, **Query** Q, **MsgPath** MP, **int** k, **float** t_{greedy}, **float** $rand$, **Set** $\mathcal{SC}$, **int** $maxTTL$, **boolean** $queryRelaxation$
Ensure: $|MP| <$ *maxTTL*
1: **Set** $resourceDependentShortcuts := \mathcal{SC}.resourceDependentShortcuts$
2: **Set** $bootstrappingShortcuts := \mathcal{SC}.bootstrappingShortcuts$
3: **Set** $defaultNetworkShortcuts := \mathcal{SC}.defaultNetworkShortcuts$
4: **Queue** *selectedPeers* := resourcePeerSelection(O, Q, k, t_{greedy}, $resourceDependentShortcuts, queryRelaxation$)
5: **if** (|*selectedPeers*| $< k$) **then**
6: *selectedPeers*.append(topBoot($bootstrappingShortcuts, (k - |s|)$)
7: **end if**
8: *selectedPeers*.append(randomFill($defaultNetworkShortcuts, rand, k$))
9: *selectedPeers* :=removeAlreadyVisitedPeers(*selectedPeers*, MP)
10: **Return** *selectedPeers*.

Algorithm 1 defines the basic peer selection procedure for choosing k peers: First it selects at most *k* peers from content or recommender shortcuts that match the resources of the query (*cf.* Line 4). Depending on the relaxation mechanism we either rank shortcuts according to their similarity with the query or we apply a query relaxation procedure. To avoid forwarding queries along shortcuts with only low similarity a minimum similarity threshold t_{greedy} is required to hold between the resource(s) of the query and the shortcut. If less than *k* shortcuts have been found, the algorithm selects the top bootstrapping shortcuts (*cf.* Line 6). Finally, remaining slots for query forwarding are filled by a random selection from the default network (*cf.* Line 8). The algorithm is not invoked if the query has reached its maximum number of hops. Furthermore, the algorithm is constrained such that a query is not forwarded to a peer if this peer has already occurred in the message path of the query (*cf.* Line 9).

Algorithm 2 allows for selecting the top peers depending on resource specific shortcuts. Independently of the relaxation strategy the query is first split into its atomic triple queries

[10] 'short' and 'long distance' as seen from the default underlying network.

(*cf.* Line 1). The resource specific peer selection algorithms are applied for all distinct triple queries and result in a selection of peers for each of them. The algorithm using query relaxation techniques for peer selection is described in more detail in section 6.4.2.1 while section 6.4.2.2 elaborates on peer selection based on semantic similarities between resources. The results for each triple query are combined according to the function defined in Section 6.4.2.3 (*cf.* Line 12) and ranked (*cf.* Line 13). Only the k best ranking peers are returned for further processing.

Algorithm 3 selects the peers with highest locally known bootstrapping capability (*cf.* Line 1). The local index contains only bootstrapping shortcuts to peers for which resource depended shortcuts are known. The function `rankBootstrapping` orders the available peers according to their bootstrapping capability calculated according to Equation 6.2. Moreover, the selection as bootstrapping peer depends on the local peers own bootstrapping capability. If a remote peer passes the bootstrapping threshold, *i.e.*, has at least twice the bootstrapping capability of the local peer it can qualify as bootstrapping peer (*cf.* Line 4). The factor was determined in experiments. Only the highest k are returned for further processing (*cf.* Line 7).

The task of algorithm `randomFill` is twofold. On the one hand it chooses from the default network layer a number of peers in order to ensure that any query can be sent to *k* different peers. On the other hand the algorithm prevents overfitting of the selec-

Algorithm 2 Resource Dependent Peer Selection:
resourcePeerSelection$(O, Q, k, t_{greedy}, resourceDependentShortcuts, queryRelaxation)$

Require: **LocalNodeRepository** O, **Query** Q, **int** k, **float** t_{greedy},
Set $resourceDependentShortcuts$, **boolean** $queryRelaxation$

1: **Set** $\mathcal{TQ}$:=extractTripleQuery(Q)
2: **Set** $selectedResourcePeers := \emptyset$
3: **Queue** *selectedPeers* $:= \emptyset$
4: **for all** $TQ \in \mathcal{TQ}$ **do**
5: **URI** $queryResource$:=extractQueryResource(TQ)
6: **if** $queryRelaxation$ **then**
7: $selectedResourcePeers$:=relaxationBasedPeerSelection$(O, queryResource, TQ, k, t_{greedy}, resourceDependentShortcuts, selectedResourcePeers)$;
8: **else**
9: $selectedResourcePeers$:=similarityBasedPeerSelection$(O, queryResource, TQ, k, t_{greedy}, resourceDependentShortcuts)$;
10: **end if**
11: **end for**
12: *selectedPeers* :=calculateMultiplication$(selectedResourcePeers)$
13: *selectedPeers* :=rankSelectedPeers$(selectedPeers)$
14: **return** selectTopKPeers$(selectedPeers, k)$

Algorithm 3 Bootstrapping Peer Selection: topBoot($bootstrappingShortcuts, k$)

Require: **Set** $bootstrappingShortcuts$, **int** k

1: **Queue** *selectedPeers* :=rankBootstrapping($bootstrappingShortcuts$)
2: **for all** $peer \in$ *selectedPeers* **do**
3: **if** $peer.bootStrappingCapability < 2 * localPeer.bootStrappingCapability$ **then**
4: *selectedPeers*.remove($peer$)
5: **end if**
6: **end for**
7: **return** selectTopKPeers(*selectedPeers*, k)

tion process[11] by exchanging selected peers with randomly chosen ones depending on the probability $rand$.

Algorithm 4 Random Peer Selection:
randomFill(*selectedPeers*, $defaultNetworkShortcuts, rand, k$)

Require: **Queue** *selectedPeers*, **Set** $defaultNetworkShortcuts$, **float** $rand$, **int** k

1: **Queue** $postSelectedPeers := \emptyset$
2: **Set** $tmpNetwork := defaultNetworkShortcuts$
3: **while not** (*selectedPeers* equals $\emptyset$) **do**
4: **Peer** $Next :=$ *selectedPeers*.pop()
5: **if** $rand(0,1) > rand$ **then**
6: $postSelectedPeers$.push($Next$)
7: **end if**
8: **end while**
9: **int** $k := k - |postSelectedPeers|$
10: **while** $k > 0$ **do**
11: $postSelectedPeers$.push($tmpNetwork$.pop())
12: $k := k - 1$
13: **end while**
14: **Return** $postSelectedPeers$

Example In our example peer 2 selects two peers for a query regarding /Topic/-TourismTechnology/BookingSystem ∧ /Topic/TourismTechnology/GeographicalInformationSystem. [12]

[11] Inspired by experiences in the field of simulated annealing and other optimization techniques (Kirkpatrick et al., 1983).

[12] The respective SeRQL query can be formulated as: `construct * from {instance} <!http://www.w3.org/1999/02/22-rdf-syntax-ns#type> {<!http://swap.ibit#BookingSystem>}; <!http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`

In the first step the local shortcuts are extracted from the shortcut index (*cf.* Algorithm 1 line 1 *ff.*). The resource dependent shortcuts are listed in Table 6.6 . Bootstrapping shortcuts are available for peer 3 and 8 while peer 2 is connected to peer 1 and 4 on the default network.
In Algorithm 2 the query is split into two triple queries namely
(*,`rdf:type`,`http://swap.ibit#BookingSystem`) and
(*,`rdf:type`,`http://swap.ibit#GeographicalInformationSystem`);
both triple queries are evaluated in either the relaxation based or similarity based peer selection algorithm. The application of the relaxation based algorithm with a value for $t_{greedy} = 0,4$ results in the selection of peer 8 (1)[13] for the first and second resource. The application of the similarity based algorithm with a value for $t_{greedy} = 0,4$ results in the selection of peers 8 (1) and 5 (100) for both query resources. The number of statements of the selected peers are combined and the resource based selection algorithm returns peer 8 for the query relaxation based algorithm and peers 5 and 8 for the similarity based algorithm.

In the query relaxation based peer selection case the peer list contains only one peer, thus the bootstrapping peer selection algorithm is invoked. Only peer 3 has twice peer 2's Bo value. Peer 3 is appended to the list of selected peers.

In the similarity based peer selection case the peer list contains two peers, thus the bootstrapping algorithm is not invoked. According to Algorithm 4 the selected peers can be exchanged with a certain probability for peers from the default network layer. In case no peers are exchanged, the query is sent to peer 8 and 3 in the query relaxation case and peer 5 and 8 in similarity case. The receiving peers answer the query and forward it following the same procedure as peer 2. In our network only peer 8 is able to answer the query. On the arrival of that answer peer 2 updates its shortcut index (*cf.* Table 6.7).

Pid	Resource	Query Hits	Maxsim	Type	Update	Rank
5	.../TourismActivity/TravelDistribution	100	1	c	245 sec	1
3	.../TourismActivity/TravelDistribution	10	1	c	240 sec	2
3	.../TourismActivity/ DestinationManagement	1	0,63	r	140 sec	3
8	.../TourismTechnology/BookingSystem	10	0,42	c	0 sec	4
8	.../TourismTechnology/ GeographicalInformationSystem	10	0,42	c	0 sec	5
1	.../TourismActivity/TravelDistribution	0	1	c	219 sec	6
4	.../TourismActivity/TravelDistribution	0	1	c	220 sec	7

Table 6.7.: Example: Shortcut Index of Peer 2 after the Response from Peer 8

{<!`http://swap.ibit#GeographicalInformationSystem`>}
[13]Number in brackets represents the number of statements

6.4.2.1. Query Relaxation Based Peer Selection Algorithm

Relaxation of a query can be achieved in a number of ways. We exploit the following considerations, which are also summarized in Table 6.8. The relaxation order corresponds to the shortcut creation and evaluation strategy and gives more value to subject and object part of a triple query than on the predicate.

1. A peer may be knowledgeable about a particular query, if one knows he had statements about the same subject-object combination, but with other predicates (states 1-3).

2. A peer may be knowledgeable about a subject-predicate combination, if one knows he had statements about the same subject alone, but maybe with other predicates (state 4).

3. A peer may be knowledgeable about an object, if one knows he had statements about the object, but where the object appeared in the subject position (state 5).

4. A peer may be knowledgeable about a property, if one knows he had statements about the superproperty (state 6).

5. A peer may be knowledgeable about a subject, if either

   a) the subject is a class and one knows the peer was knowledgeable about the superclass (state 7a), or

   b) the subject is an instance and one knows the peer was knowledgeable about a class the subject is an immediate instance of (state 7b).

6. A query for everything or the $\mathcal{ROOT}$ concepts or properties (i.e. in RDF these are rdf:resource, rdfs:class, rdfs:property, rdf:type, etc.) cannot be relaxed further.

Algorithm 6 implements these considerations. It exploits Table 6.8 in order to derive a(n often single-element) set of relaxed queries (*cf.* Line 16 in Algorithm 5). One may note in particular: *(i)* multiple relax queries exist when one asks for super properties of a given property or immediate types of a subject; *(ii)* implicitly this relaxation is recursively applied in Algorithm 5 (*cf.* Line 20); and, *(iii)* there remain other options for query relaxation; first tests revealed that the above ones give quite good results for later-on selecting an appropriate peer to send the original query to. Other query formalisms such SQL require different query relaxation techniques (*cf. e.g.*, (Chu et al., 1996; Motro, 1990)).

The average number of statements for different shortcuts from the same peer is calculated in the method `rankPeers` (*cf.* Line 18) and the peers are sorted according to the resulting number of statements.

State	Query	Relaxed Query
1	(s, p, o)	$(s, *, o)$
2	$(s, p, *)$	$(s, *, *)$
3	$(*, p, o)$	$(*, *, o)$
4	$(s, *, o)$	$(s, *, *)$
5	$(*, *, o)$	$(o, *, *)$
6	$(*, p, *)$	$(*, super(p), *)$
7a	$(s, *, *)$	$(super(s), *, *)$
7b	$(s, *, *)$	$(class(s), *, *)$
8	$(*, *, *) \vee (\mathcal{ROOT}, *, *) \vee (*, \mathcal{ROOT}, *)$	

Table 6.8.: Query Relaxation Order

Example The query (*,rdf:type,http://swap.ibit#BookingSystem) is first relaxed to (*,*,http://swap.ibit#BookingSystem) and evaluated, and then to (http://swap.ibit#BookingSystem,*,*) before the first shortcut can be found. Peer 8 is appended to the list of selected peers. The query is further relaxed to (http://swap.ibit#Topic,*,*) in order to select enough shortcuts for query routing. The similarity between Topic and the query resource is 0.35, thus below the threshold leading to the termination of the algorithm. The same procedure is followed for the query (*,rdf:type,http://swap.ibit#GeographicalInformationSystem).

6.4.2.2. Semantic Similarity Based Peer Selection Algorithm

Exploiting a shared ontology to select promising remote peer for query forwarding is also the idea of approaches based on semantic similarities between a query and indexed shortcuts. Algorithm 7 uses a similarity function between a triple query q and a shortcut sc, which are both given by query terms in the same ontology with the following property: $sim : q \times sc \rightarrow [0; 1]$. Such similarity metrics are often domain specific and depend on the query semantics. In our implementation we use a similarity metric for topic hierarchies proposed by Li et al. (2003) (but of course any other suitable similarity measure can be used, *e.g.*, *cf.* (Ehrig, 2006)):

$$sim_{Topic}(q, sc) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } q \neq sc \\ 1 & \text{otherwise} \end{cases} \tag{6.3}$$

where l is the length of the shortest path between q and sc in the graph spanned by the sub concept relation and h is the minimal level in the ontology of either q or sc. α and β are parameters scaling the contribution of shortest path length l and depth h, respectively. Based on the benchmark data set given in Li et al. (2003), we chose $\alpha = 0.2$ and $\beta = 0.6$ as optimal values.

Algorithm 5 Query Relaxation Based Peer Selection:
relaxationBasedPeerSelection($O, queryResource, \mathcal{TQ}, k, t_{greedy}$, $resourceDependentShortcuts$, *selectedPeers*)

Require: **LocalNodeRepository** O, **URI** queryResource, **Set** $\mathcal{TQ}$, **int** k, **float** t_{greedy}, **Set** $resourceDependentShortcuts$, **Queue** *selectedPeers*

1: $\mathcal{TQ}_{relaxed} := \emptyset$
2: **Set** selectedResourcePeers:= $\emptyset$
3: **for all** $TQ \in \mathcal{TQ}$ **do**
4: **URI** $relaxedResource$:=extractQueryResource(TQ)
5: **if** sim($relaxedResource, queryResource$) $> t_{greedy}$ **then**
6: S_Q :=performQuery(O, TQ)
7: **for all** $S \in \mathcal{S}_Q$ **do**
8: **URI** $answerResource$:=extractAnswerResource(S)
9: **for all** $sc \in resourceDependentShortcuts$ **do**
10: **if** sc.resource = $answerResource$ **then**
11: $selectedResourcePeers$.push(pid, sc)
12: **end if**
13: **end for**
14: **end for**
15: **end if**
16: $\mathcal{TQ}_{relaxed} := \mathcal{TQ}_{relaxed}$+relaxQuery($O, TQ$)
17: **end for**
18: *selectedPeers* := *selectedPeers*.append(rankPeers($selectedResourcePeers$))
19: **if** $|$*selectedPeers*$| < k$ **then**
20: *selectedPeers* :=relaxationBasedPeerSelection($O, queryResource, \mathcal{TQ}_{relaxed}$, $k, t_{greedy}, resourceDependentShortcuts$, *selectedPeers*)
21: **end if**
22: **return** *selectedPeers*

Algorithm 6 Query Relaxation: relaxQuery(O, TQ)

Require: O, TQ

1: *state* := determineState(TQ)
2: $\mathcal{TQ}$:= newQuery(O, TQ, *state*)
3: **return** $\mathcal{TQ}$

The Algorithm 7 makes use of the semantic similarity between the shortcuts and the query resources in two different ways. First, only shortcuts above the similarity threshold t_{greedy} are selected for further processing. Second, the selected shortcuts are ranked taking into account the similarity with the query resource. The method `sortSimCSRS` sorts the shortcuts according to their similarity with the query resource (*cf.* Line 7). In case of equal similarity, the number of content statements is considered. If this ranking does not produce a complete ordering of the shortcuts, no further order operation is performed. Only the k

Algorithm 7 Similarity Based Peer Selection:
similarityPeerSelection($O, queryResource, TQ, k, t_{greedy}, resourceDependentShortcuts$)

Require: LocalNodeRepository O, **URI** $queryResource$, **Query** TQ, **int** k, **float** t_{greedy},
Set $resourceDependentShortcuts$

```
1:  Queue selectedResourcePeers := ∅
2:  for all sc ∈ resourceDependentShortcuts do
3:    if sim(sc, queryResource) > t_greedy then
4:      selectedResourcePeers.push(pid,sim(sc, queryResource), sc)
5:    end if
6:  end for
7:  selectedResourcePeers.sortSimCSRS()
8:  int i := 0
9:  while i <= k do
10:   (pid,sim(sc, queryResource), sc) := selectedResourcePeers.pop()
11:   selectedPeers.push(pid, sc)
12:   i := i + 1
13: end while
14: return selectedPeers
```

best ranked peers are returned (*cf.* Line 9 ff.).

Example The similarities between . . . /BookingSystem and . . . /GeographicalInformationSystem and the available shortcuts are calculated. This results in the selection of peer 8 with the shortcut . . . /BookingSystem[14] and peer 5 with the shortcut . . . /TravelDistribution[15] for the first and second query resource.

6.4.2.3. Conjunctive Queries

Each query may be composed of several triple queries, *e.g.*, *Select all resources that belong to the topic GeographicalInformationSystem and to the topic DestinationManagement.* In our context we formalize this query using common ontologies: *Find any resource with the topics Topic/TourismTechnolgy/GeographicalInformationSystem* $\wedge$ */Topic/-TourismTechnology/DestinationManagement.* A default approach would route a query only to a remote peer for which shortcuts for *all* triple queries of the query are stored using an exact match paradigm. Too specific query predicates under the exact match paradigm often lead to empty result sets and do not appropriately consider negation. The notion of best matches and relative importance of predicates can be a good alternative to satisfy users information needs independently of the individual peer instances. In Tempich et al.

[14] Similarity with the first query resource 1 and the second 0,42.
[15] Similarity with the first query resource 0,63 and the second 0,42.

(2005b) we investigated metrics to determine the best peers to route a query using multi predicate queries in shortcut networks. We observed satisfying results using the selection function described in Cooper (2004) which uses an equation similar to Equation 6.4 to combine query hits for distributed document retrieval. We refer to this strategy as *Multiply*.

$$R_p(q) = \prod_{i=1}^{\#t} q_i^p \tag{6.4}$$

We calculate the relevance R for a peer p for a query q using Equation 6.4, where $\#t$ represents the number of resources in the query, q_i^p represents the query hits per resource i of each peer matching at least one of the resources of the query. We select the peers with the highest relevance. Equation 6.4 is executed in Algorithm 1 in the method `calculateMultiplication` (*cf.* Line 12).

6.5. Summary

REMINDIN' is a query routing algorithm for pure and unstructured P2P networks. It meets the requirements on query routing of a distributed knowledge management scenario. REMINDIN' imitates social metaphors, observed in human social networks, in order to select among a number of available peers the most appropriate ones. A peer harvests knowledge about remote peers if queries are forwarded to it for query processing. REMINDIN' organizes the collected information in different semantic overlay layers by creating non-forwarding indexing links, *i.e.* shortcuts. It maintains content provider, recommender and bootstrapping shortcuts on top of the default network layer in a shortcut index. In order to maintain a small shortcut index REMINDIN' retains only shortcuts which are up to date and semantical similar to the local content. Appropriate peers to send or forward a complex query to are selected by comparing the query with the shortcuts. If REMINDIN' cannot find an exact match between a query and a local shortcut the ontology is utilized to find similar shortcuts and thus remote peers which might be able to answer the query. REMINDIN' implements relaxation strategies and similarity measures based on RDF(S) ontologies. In that REMINDIN' uses only local information to select remote peers. This enables the user to decide on a per-query basis if he wants to share knowledge with remote peers.

The simulations presented in the next Chapter demonstrate, that peers using REMINDIN' find a high proportion of the knowledge available in the network. REMINDIN' achieves this in static as well as volatile networks. The limitation of the index size does not considerably influence the performance.

7. Evaluation of REMINDIN'

Overview

The evaluation of the REMINDIN' algorithms is based on simulations of a semantic peer-to-peer network. We developed our own simulation environment. The components in the simulation are the same as for the SWAP system except the communication layer which is based on an discrete event simulator. The discussion of the evaluation results begins with the selection of evaluation criteria followed by the description of the evaluation data sets. We list a number of hypothesis before we present the results of the simulation.

References: This chapter is based on the publications (Ehrig et al., 2003), (Tempich et al., 2004b), (Löser et al., 2005b), (Löser et al., 2005a), and (Tempich et al., 2005b).

7.1. Evaluation Criteria for Routing Algorithms

The evaluation criteria for peer-to-peer (P2P) routing algorithms depend on the requirements of the application scenario. We recall that the requirements are structured according to the dimensions Expressiveness (1), Comprehensiveness (2), Autonomy (3), Efficiency (4), Quality of service (5), Robustness (6) and Security (7) (*cf.* Section 6.1.2). Requirements 1, 2, 6 and 7 determine the architecture of the P2P system while requirements 3, 4 and 5 are subject to evaluation. In following we define the evaluation measures in order to determine the level of compliance with requirements 3 to 5. The remaining requirements are the source for the hypothesis formulated in Section 7.3.

The efficiency and quality of service are measured in terms of recall and messages per query.

Recall describes the proportion between all relevant answers in peer network and the retrieved ones. Hereby, we defined 'relevant' as 'matches the query'. We imply that a high recall corresponds to a high quality of service. We do not rank the answers and assume that all of them are equally relevant to the query. The recall is calculated according to Equation 7.1.

$$R = \frac{|retrieved|}{|relevant|} \tag{7.1}$$

Messages per query (M_q) represent the required search costs. This criteria is used to determine the efficiency of the routing algorithm. The less messages the algorithm produces for one query the more efficient it is.

Message Gain (MG_q) is defined as recall per message. We use the message gain in some charts to visualize the trade-off between recall and number of messages. The message gain is calculated according to Equation 7.2.

$$MG_q = \frac{R}{M_q} \tag{7.2}$$

Clustering coefficient and average path length are measures to evaluate the small-world characteristics of a network (Albert & Barabási, 2002). Small-world networks are characterized by a high clustering coefficient and a low average path length.[1] These characteristics are on the one hand associated with efficient search results (Adamic et al., 2001) and a high fault tolerance against random node failures (Hong, 2001). On the other hand, the clustering coefficient indicates that peers with similar interests are close to each other in the network. In our knowledge management scenarios it is desirable that people who share common interests get to know each other. Hence, the design of the routing algorithm allows for a high peer autonomy yet if the clustering coefficient reaches a high level it still supports collaboration.

Clustering coefficient The clustering coefficient represents the compactness of the network. It captures how many of the neighbors of a node are connected to each other. The clustering coefficient is defined according to Equation 7.3.

$$\mathcal{C} = \frac{1}{|V|} \sum_{v \in V} \frac{E(\Gamma_v)}{k_v * (k_v - 1)} \tag{7.3}$$

V denotes the set of peers in the network, k_v denotes the maximum number of shortcuts for a peer v, Γ_v the direct neighbors of a peer and $E(\Gamma_v)$ represents a function that counts the number of links in Γ_v.

Average path length A short average path length denotes a highly directed information flow between two peers in the network. Given two arbitrary selected peers $v_1, v_2 \in V$ and $d_{min}(v_1, v_2)$ the minimum path length between v_1 and v_2, the average path length is defined as

$$d = \frac{1}{\binom{|V|}{2}} \sum_{v_1 \neq v_2} d_{min}(v_1, v_2) \tag{7.4}$$

[1] The clustering coefficient is high in comparison to a random network. A random network has a low average path length and a low clustering coefficient. The average path length is low in comparison to a network organized in a ring. A ring has a high clustering coefficient and a long average path length.

7.2. Evaluation Setting

Evaluation of the routing algorithms is performed in a multi stage process. First, we have selected three different data sets. A data set represents the total knowledge available in the simulated P2P network. Second, the complete data set is divided into possibility overlapping chunks of knowledge. These chunks setup the peer local knowledge. Third, from a data set we generate queries. The statements retrieved if a query is evaluated against the complete data set determines the number of relevant answers. Forth, each peer is assigned a number of queries which the peer submits to the network. Fifth, the peers are arranged in a default network topology before the simulation is started. Sixth, during the simulation we collect data to calculate the evaluation measures. Finally, the simulations are repeated with different parameter settings to demonstrate the influences of single parameters. In the remainder we describe the data sets, the content distribution and the queries and their distribution.

7.2.1. Evaluation Data Sets

Three data sets are used as content in the P2P network simulation. The Bibster data set was generated from real world observations of the Bibster system. The DMOZ data set is an extract from a real Web site catalog. The synthetic data sets were generated using different generation parameters.

7.2.1.1. The Bibster Data Set

This data set bases on real query data captured from the P2P bibliography network Bibster (*cf.* Section 3.1.2, (Haase et al., 2004a)). We use data from observations in a four month period. In this time 520 peers were online. As we logged only the queries and the number of retrieved answers not all shared BibTEX items are available for the simulation setup. From the answers and queries we constructed a data set containing in total 26.173 distinct BibTEX items and 37 distinct classes. The items are classified against the different topics available from an ACM-topic hierarchy[2].

7.2.1.2. The DMOZ Data Set

Participants of the open directory project (DMOZ) manually categorize Web pages of general interest into a topic hierarchy. Editors contribute links to Web pages, define subtopics and associate related topics to the DMOZ topic pages. The DMOZ data is available as an RDF dump comprising a small schema and many instances. The main classes of the DMOZ data set are Topic, Alias, and ExternalPage:

[2]The RDF serialization of the ACM-topic hierarchy is available online at:
`http://www.aifb.uni-karlsruhe.de/WBS/pha/bib/acmtopics.rdf`

Topic The resource Topic represents a topic in the DMOZ hierarchy and has properties for link[3], containing a reference to an ExternalPage and to an editor, viz. an editor of the topic. The properties related, symbolic and narrow describe relations to other topics and aliases.

Alias The resource Alias represents a topic with a similar meaning as the topic and has properties for Title and Target. Target is the relation pointing to the similar Topic.

ExternalPage The resource ExternalPage represents URIs to Web pages and has the properties Title and Description

The data source has some interesting properties rendering it adequate for our evaluation purposes: (1) There are many relations other than taxonomic ones between the topics — in contrast to many other datasets. (2) Topics have editors, many have several — implying a natural way to assign statements to peers. (3) Topics are 'populated' with many links.

The DMOZ hierarchy is available in pure RDF only. To enhance its semantic description, we have converted it to RDFS. We converted the topics to instances of `rdfs:class`. narrow and narrow1 were converted into `rdfs:subClassOf`. The properties link were interpreted as `rdf:type` of the topics they belong to. For instance, `http://www.w3.org/People/Berners-Lee/` is then an instance of `http://dmoz.org/Computers/Internet/History/People/Berners-Lee,_Tim/`.

DMOZ	RDF(S) entity	Example	Number
Schema			
Topic	Class	Top/Arts	1657
Property	Property	symbolic, related	16
Instances			
link	rdf:type	"`http://www.w3.org/People/Berners-Lee/`" rdf:type "Top/Computers/Internet/"	45347
symbolic, related	symbolic, related	"Top/Arts/" related "Top/Business/Arts_and_Entertainment"	3520
narrow, narrow1	subClassOf	"Top/Arts/Movies" subClassOf "Top/Arts"	1952
editor	Peer		1024

Table 7.1.: Survey of DMOZ Open Directory Structure Realized as RDF Schema and Instances

In order to handle the sheer size of the DMOZ hierarchy, we included only the first three levels of the hierarchy in our experiments. Table 7.1 summarizes the properties of the data set used in the simulation.

[3]DMOZ distinguishes important and less important links by categorization into link, link1, link2. We have merged these three again into one property. The analogous case is true for symbolic, etc.

7.2.1.3. Synthetic Data Sets

The data in the Bibster as well as the DMOZ data set is distributed manually by humans working with the respective systems. In order to explore the influence of data distribution parameters on REMINDIN' several synthetic data sets are created. We describe the properties and settings for the generation of synthetic data sets for semantic P2P simulations. Such a data set comprises an instantiated ontology formalizing the complete knowledge available in the network, and an assignment of knowledge to peers in the network. The number of classes, the number of properties and the number of sub-class relationships together with their respective distributions determine the schema of the ontology. The number of instances and the number of relations between instances determine the distribution of the instance data. The distributions are modeled as Zipf distributions with parameter settings according to observations from real world data sets. The parameter settings for the schema generation are based on Tempich & Volz (2003) while the parameter settings for instance generation are based on observation of Cholvi et al. (2004). Data distribution on the peers follow the model presented in (Crespo & Garcia-Molina, 2002b).

For the schema we choose to generate an ontology with 1.000 classes and assign a popularity based on a Zipf distribution with skew factor 1 to each class. The popularity of a class influences the number of instances, the replication in the network and the connectivity with other classes through properties. We select the number of sub-classes and the number of properties of a class according to a Zipf distribution with the skew factor 1.1. This results in 357 properties.

200.000 instances are generated and assigned to classes based on the popularity of the different classes; each instance is assigned to one class. A $TotalNoPropertyInstances = 100 * TotalNoProperties$ of properties between instances are generated. Likewise to classes, a property schema (one could also call it a binary relation), has a popularity based on a Zipf distribution with skew factor 1 that is considered when generating properties between instances (*i.e.* when populating the binary relations).

7.2.2. Content Distribution

The described data sets are distributed and represent the local knowledge of the peers. The goal of the knowledge distribution is to resemble the knowledge distribution in a real world network. The next sections elaborate on the chosen distribution parameters.

7.2.2.1. Content Distribution of the Bibster Data Set

The BibTEX items are assigned to peers following the observations on the real network. Each peer is assigned the items which it has sent as answers to queries. The Bibster data can thus be distributed in the same way as it has been in the real world setting. In average

each peer shared 50 items and provided data for 1 class. However, the distribution is highly skewed as visualized in Figures 7.1 and 7.2, *e.g.*, 328 peers do not share any knowledge while one peer contributes 5% of all BibTEX items and covers nearly 30% of the classes.

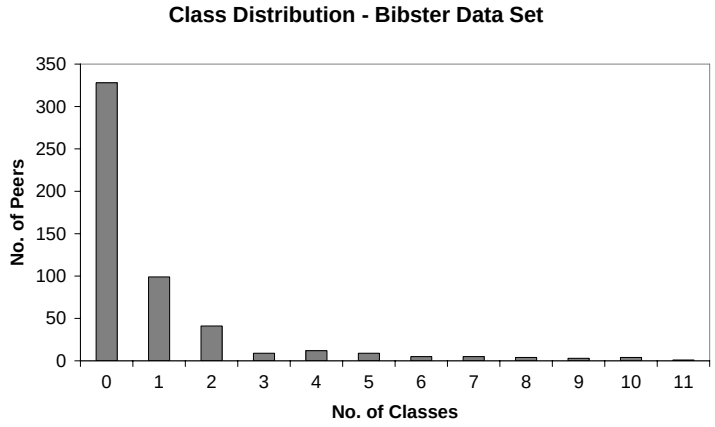


Figure 7.1.: Bibster Data Set: Class Distribution on Peers

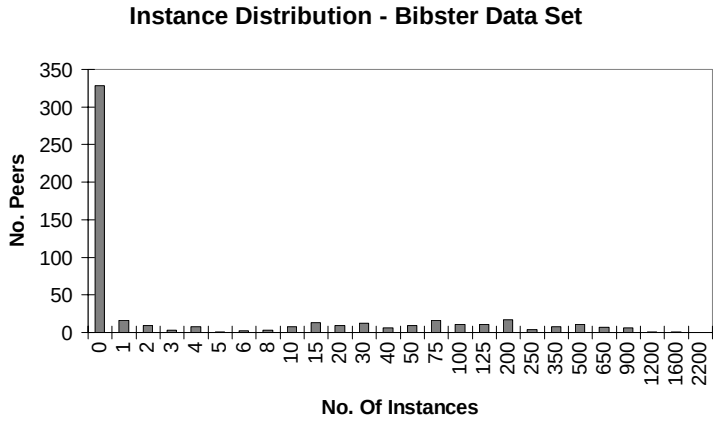


Figure 7.2.: Bibster Data Set: Instance Distribution on Peers

7.2.2.2. Content Distribution of the DMOZ Data Set

All of the 1657 topics in the first three levels of the DMOZ hierarchy have one or more editors assigned to them. Everybody can become an editor of a category in DMOZ. DMOZ encourages users to "choose a topic you know something about and join". Hence, we assume that editors who edit a topic (which became classes in RDF(S)) also locally store links they have assigned to a topic. Since a topic can have more than one editor not all of

the links need to come from one editor alone. Finally, editors may also add links to other topics. Thus, they are probably also informed about related topics but to a lesser extent.

These assumptions led us to the following distribution of instances in our simulation. We represent one editor by one peer, thus we have 1024 peers. Assuming that an editor is not the source of all instances within 'his' topic ('his' class) we choose randomly 70% of the direct instances within 'his' class and assign them to the Local Node Repository (LNR) of the peer. In addition, we consider all classes directly related to 'his' class (via subclass-of, via related, etc.) and we randomly assign 12% of the direct instances of these directly related classes to the LNR of the peer. Thus, all peers have an individual local node repository. The resulting information distribution is visualized in Figures 7.3 and 7.4.

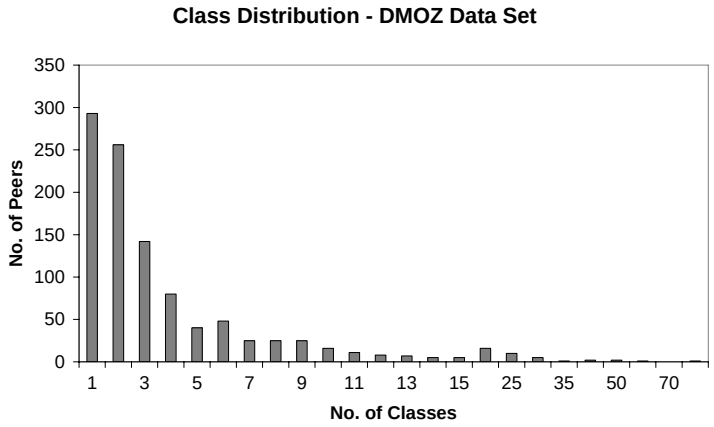


Figure 7.3.: DMOZ Data Set: Class Distribution on Peers

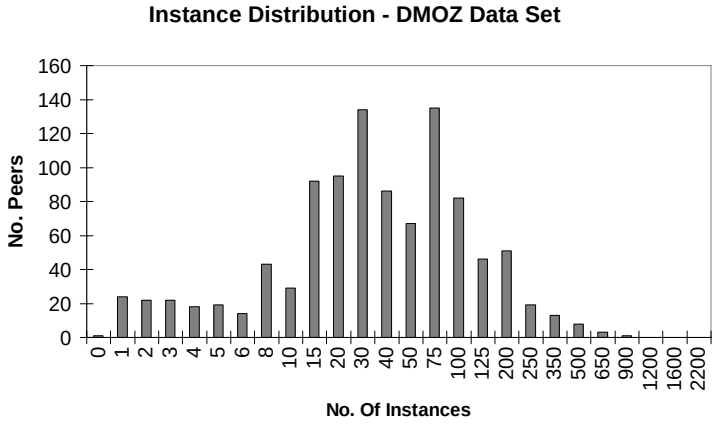


Figure 7.4.: DMOZ Data Set: Instance Distribution on Peers

7.2.2.3. Content Distribution of the Synthetic Data Sets

For the synthetic data set no 'natural' knowledge distribution over peers is available. In order to distribute the content over the peers we adopt the model presented in (Crespo & Garcia-Molina, 2002b). Assignment of data to peers is based on the assumption that users are generally interested in a small subset of the entire content available in a P2P network. We have modeled that the interests are more likely in only a limited number of content classes and thus users would be more interested in some classes while less in others. The maximum number of classes that a peer is interested in is computed as $ClassesOfInterest = ln(NoOfClasses) * 2$. The actual number of classes is chosen randomly from a uniform distribution. Observing the studies in Cholvi et al. (2004) neither do all peers share the same amount of data and nor exhibit the same 'social behavior'. For instance, a large number of users are so-called 'free riders' or freeloaders who do not contribute anything to the network but essentially behave like clients. On the other hand, a small number of users (less than 5%) provide more than two thirds of the totally available amount of data and thus behave like servers. Considering the study in Adar & Huberman (2000) the following storage capacity is assigned to the peers in the network: 70% of the peers do not share any instances (free riders); 20% share 100 instances or less; 7% share 101 up to 1000 instances and finally, only 3% of the peers share between 1001 and 2000 instances. Instances are chosen from the classes the user is interested in. The resulting information distribution is visualized in Figures 7.5 and 7.6. A peer sharing an instance knows all its properties and the type of the range instance.

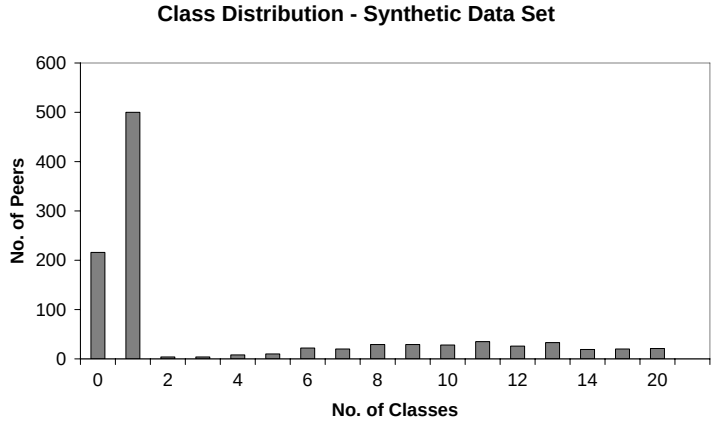


Figure 7.5.: Synthetic Data Set: Class Distribution on Peers

7.2.3. Queries and Query Distribution

Besides the content distribution the assignment of queries to peers influences the evaluation results of the algorithms. The assignment of queries to peers should resemble real world

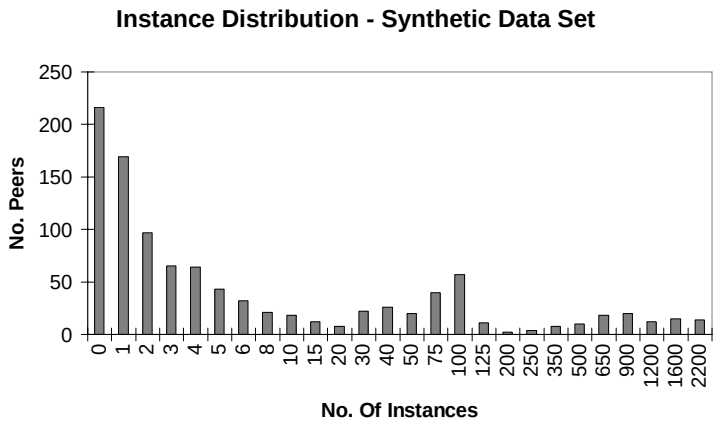


Figure 7.6.: Synthetic Data Set: Instance Distribution on Peers

querying behavior. We describe for the three data sets the assignment of queries to peers.

7.2.3.1. Queries for the Bibster Data Set

For the Bibster data set we know from our observations the originating peers of the queries. In total 398 peers issued 3319 queries. From these queries 1949 contained 1 predicate, 1180 contained 2, 180 contained 3, 7 contained 4, 2 contained 5 and 1 contained 6. A query containing two attributes looks, *e.g.*, like

```
construct * from
{} <!http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
{<!http://www.semanticweb.org/ontologies/swrc-onto-2001-12-11.daml#
Publication>};
<!http://www.semanticweb.org/ontologies/swrc-onto-2001-12-11.daml#
year>
{name1}
where name1 like "2003";
```

In the simulation a peer issues the same queries as during the observation period. In order to scrutinize the effects of a longer experiment a query may be repeated in the simulation.

7.2.3.2. Queries for the DMOZ Data Set

Queries are generated in the experiments by instantiating the blueprint $(*, < rdf : type >, topic)$, with $topics$ arbitrarily chosen from the set of topics that had at least one instance. We generated 1657 different queries of the following type.

```
construct * from
{} <!http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
{<!http://dmoz.org/Top/Computers/Internet/Searching>}
```

In the simulation the peers issue 30.000 queries uniformly distributed over the 1657 generated queries. We choose a uniform query distribution instead of a ZIPF-distribution, typically observed in file sharing networks (Saroiu et al., 2003), thus the algorithm does not take advantage of often repeated queries for popular topics. The set of 1657 queries is split into two sets of equal size. There were two phases. First, there was a 'learning phase' where the peer network was confronted with the first set 828 queries. Then, there was an explicit 'test phase', in which one could observe how the peer network would re-adjust to the second set of queries.

7.2.3.3. Queries for the Synthetic Data Set

In order to test our algorithm we have generated two different types of queries. All queries ask for instances satisfying a varying number of constraints. The first query type instantiates the blueprint $(instance; isTypeOf; class) \wedge (instance;\ property;\ instance2) \wedge (instance2; isTypeOf; class2)$. We thus query for all $instance$s of a certain $class$ with the constraint that the instance has one particular $property$ pointing to another instance. The range of the $property$ determines the type of $instance2$. From this type of queries we generated 2194. The second query type extends the first one by adding a constraint on the properties and instantiates the blueprint $(instance; isTypeOf; class) \wedge\ (instance; property; instance2) \wedge (instance2;\ isTypeOf; class2) \wedge (instance; property2; instance3) \wedge\ (instance3; isTypeOf; class3)$. From this type of queries we generated 1087. Such a query is visualized in the example.

```
construct * from
{instance} <!http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
{<!http://swap.simulation#class28>};
<!http://swap.simulation#property70>   {}   <!http://www.w3.org/1999/02/
22-rdf-syntax-ns#type> {<!http://swap.simulation#class3>},
{instance} <!http://swap.simulation#property72> {} <!http://www.w3.org/
1999/02/22-rdf-syntax-ns#type> {<!http://swap.simulation#class2>}
```

Queries are chosen randomly from all available queries. We choose a uniform query distribution.

7.2.4. Configuration of the Simulation

We setup all simulations with the same initial network topology. In experiments looking at the effects of a volatile network we follow a well defined strategy to determine the online and offline times of a peer.

7.2.4.1. Gnutella Style Network

The simulation is initialized with a network topology which resembles the small-world properties of file sharing networks as found in Gnutella[4]. We simulated 1024 peers and connect each peer with 5 remote peers on the network layer. In the simulation a peer can establish arbitrary connections to remote peers, if it knows the network identifier of the peer. The selection of peers for query issuing is randomized with a uniform distribution over peers. For the DMOZ and the synthetic data set the queries are chosen randomly from the available queries with a uniform distribution. As mentioned above for the Bibster data set we simulate the real world query distribution. The peers decide on the basis of their local information which remote peers to send the query to. Within one simulation peers use the same routing algorithm to select up to $k = 2$ peers to send the query to. Each query is forwarded until the maximal number of hops $h_{max} = 6$ is reached.

7.2.4.2. Volatile Network Characteristics

The simulation environment follows two strategies to determine if a peer is connected to the network or not. In the first set of simulations we assume a static network, *i.e.* all peers are always online. In the second round of simulations we us a dynamic network model observed for Gnutella networks by Saroiu et al. (2003): 60% of the peers have an availability of less than 20%, while 20% of the peers are available between 20 and 60% and 20 % are available more then 60%. Hence only a small fraction of peers is available more than half of the simulation time, while the majority of the peers is only online a fraction of the simulation time. The real online and offline times of peers are determined randomly at simulation runtime so that their cumulative online times resemble the distribution found in

[4]We used the Colt library http://nicewww.cern.ch/~hoschek/colt/ to generate the small-world topology

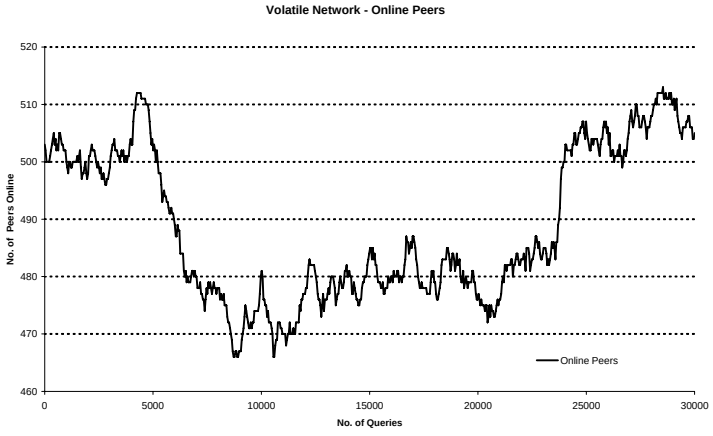


Figure 7.7.: Volatile Network: Number of Peers Online

(Saroiu et al., 2003). Figure 7.7 visualizes the number of peers online during the simulation run. A peer ensures before it forwards a query that it is connected to at least five online remote peers on the network layer. If a remote peer is offline it is exchanged with an online peer. A peer discovers with rendezvous techniques online peers on the network layer.

We also consider changing user interests. In order to simulate changing interests the peers choose the first 15 queries[5] from one half of the total queries and the second 15 queries from the remaining ones. Although studies of interest shift in file sharing networks show a smoother transition between interests, this drastic shift is better suited to exemplify the behavior of the algorithms (*cf.* (Liang et al., 2005; Christin et al., 2005)). For the Bibster data set we omit this procedure.

7.2.4.3. Simulator Setup and Simulation Statistics

We used a round based simulation framework. Table 7.2 presents the main parameters of the simulation framework. In total we simulated 1024 peers. To determine the standard error of our observations for a 95% confidence interval ($p < 0.05$) each simulation was executed eight times. We set the greedy search threshold to 0.50 and the random contribution to 0.15.

7.3. Evaluation Hypothesis

The evaluation of the REMINDIN' algorithm serves two purposes: First, we show that REMINDIN' meets the requirement defined in section 6.1.2. Second, we show that the REMINDIN' routing algorithm delivers better results w.r.t. our evaluation criteria than related approaches to routing. Before we present the results of our evaluation in the next section we formulate the hypotheses we investigate.

1. REMINDIN' is more efficient and has a higher quality of service than related routing algorithms in static networks with single triple queries. The hypothesis is validated if the recall of REMINDIN' is higher than that of related routing algorithms and the number of messages is lower.

2. REMINDIN' is more efficient and has a higher quality of service than related routing algorithms in volatile networks with single triple queries. REMINDIN' is thus more robust than related routing algorithms. The hypothesis is validated according to the same evaluation measures as hypothesis 1.

3. REMINDIN' is more efficient and has a higher quality of service than related routing algorithms in volatile networks with conjunctive triple queries. REMINDIN' thus meets the expressivity requirement. The hypothesis is validated according to the same evaluation measures as hypothesis 1.

[5] This equals to ca. 15.000 queries over all peers

Parameter	Value
Queries	30.000
Queries per peer	ca. 30
Query time to life ($maxTTL$)	6
Selected peers per query (k)	2
Greedy search threshold (t_{greedy})	0.5
Random contribution ($rand$)	15 %
Index size	40
Index parameters ($simInfluence, typeInfluence, timeInfluence$)	(0.1, 0.8, 0.1)
Relaxation based on similarity ($queryRelaxation$)	*false*
DMOZ data set	
Classes	1657
Queries before interest shift	829
Queries after interest shift	828
Simulated peers	1024
Bibster data set	
ACM topics	1293
Queries	3319
Simulated peers	520
Synthetic data set	
Classes	1000
Queries before interest shift	1641
Queries after interest shift	1640
Simulated peers	1024

Table 7.2.: Simulation Parameter Setting

4. Each layer contributes to the performance of the REMINDIN' algorithm. The hypothesis is validated if the message gain increases when applying an additional layer.

5. REMINDIN' is efficient in terms of memory consumption as a small shortcut index size is sufficient to maintain its performance levels. The hypothesis is validated if the message gain does not change for different index sizes.

6. The use of semantics for query routing is beneficial and increases REMINDIN's efficiency. The hypothesis is validated if the message gain increases if $t_{greedy} < 1$.

7. REMINDIN' is adaptable to different application areas by adjusting its parameter setting, *e.g.*, to increase the proximity between peers of similar interests. The hypothesis is validated if the clustering coefficient depends on different parameter settings.

7.4. Evaluation Results

This section presents the evaluation results of the REMINDIN' algorithm. The evaluation is organized following the order of the hypothesis formulated in the previous section. The simulations are setup with the parameter settings listed in Table 7.2. The evaluations compare REMINDIN' with related routing algorithms which fulfill the requirements of the DKM scenario. In particular the simulation environment includes implementations for flooding based routing (Kan, 2001) and Interest based locality (IBL) routing (Sripanidkulchai et al., 2003).

Kan (2001) (Naive) This routing algorithm is naive. (1) It either forwards a query to all remote peers a peer is connected to on the default network layer. This strategy results in a high recall but produces a high number of messages, because this strategy queries non-selectively most of the available peers. (2) An alternative naive strategy selects randomly from the remote peers known on the network layer a predefined number of peers k and forwards the query only to the selected peers. This strategy results in a lower recall but produces also a lower number of messages. For the purpose of clarity the figures present only the results of the later strategy. The results of the first strategy are described in the text.

Sripanidkulchai et al. (2003) (IBL) This routing algorithm stores non-forwarding indexing links to remote peers which have answered previous queries of the peer. Queries are sent on non-forwarding search links to all remote peers stored in the index. If none of the queried peers responses the query is distributed according to the second naive routing strategy. In order to compare the IBL strategy with REMINDIN' both routing algorithms use the same index size. IBL uses a least recently used strategy to maintain the index size.

Hypothesis 1 Figures 7.8 and 7.9 graphically summarize the comparison between REMINDIN', REMINDIN' without random contribution, the naive approach and the IBL strategy in a static network using the DMOZ data set. Peers issue queries for Web site links related to one topic in the DMOZ hierarchy. REMINDIN' and IBL use an index of size 40. REMINDIN' uses the relaxation based peer selection strategy ($queryRelaxation = true$). During the simulation run each peer issues in average 30 queries approximately corresponding to a 2 hour period in a real file sharing network. After 15 queries the peers shift interest and choose queries from the second half of the available ones. Figure 7.8 focuses on the recall achieved by the different routing strategies. The total number of issued queries is plotted against the achieved recall. Points in all the graphs represent averages for 1000 queries. The recall for REMINDIN' increases with the number of sent queries and levels off at about 55% recall.[6] In the event of interest shift the recall nearly halves

[6]Higher recall rates can be achieved increasing the maximum number of hops h_{max} at the cost of more messages.

but recovers after another 15 queries per peer to the same levels as before. The recall for REMINDIN' without random contribution behaves similar, but at lower levels. The IBL strategy produces recall levels of around 25%, and o around 13% after the interest shift. The recall levels off at around 20% after the interest shift. Following the naive routing strategy – randomly selecting $k = 2$ peers – results in circa 10% recall independently of the interest shift. The alternative strategy – selecting all known network peers – results in circa 60% recall at the cost of circa 600 messages per query.
Figure 7.9 plots the number of messages per query against the total number of sent queries. The number of messages per query produced by REMINDIN' decreases over time until the interest shift, when the number of messages briefly increases. It levels off again to the end of the simulation at 60. The number of messages per query is slightly higher for REMINDIN' with random contribution than without. The random contribution ensures that more remote peers may contribute to the answer. Other disciplines have also found the advantageous effects of serendipity *cf.* (Merugu et al., 2003; Ramakrishnan & Grama, 1999). The number of messages produced by the naive approach is constant at around 75. The number of messages produced by IBL first increases and levels off at around 85 messages. The interest shift does not affected the number of messages produced by the IBL strategy.

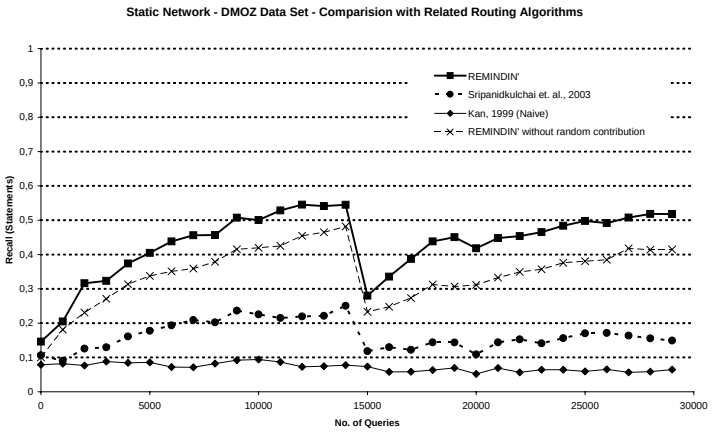


Figure 7.8.: Static Network: Comparison of Query Routing Algorithms: Recall

The simulations validate the hypothesis that REMINDIN' has a better performance than related routing approaches in static unstructured P2P networks. REMINDIN' learns quicker about the available knowledge than IBL and adapts better to interest shifts. The simulations in relation to Hypothesis 2 and 4 show that the first answer is in average found only after 3.5 hops, thus the IBL strategy is less effective and that the recommender layer contributes to REMINDIN' better performance. The simulations in relation to Hypothesis 6 demonstrate that in particular the use of query relaxation enable the algorithm to adapt quickly to interest shifts. The selection of arbitrary peers instead of the selected ones as a result of the random contribution prevents over-fitting of the algorithm at the cost of

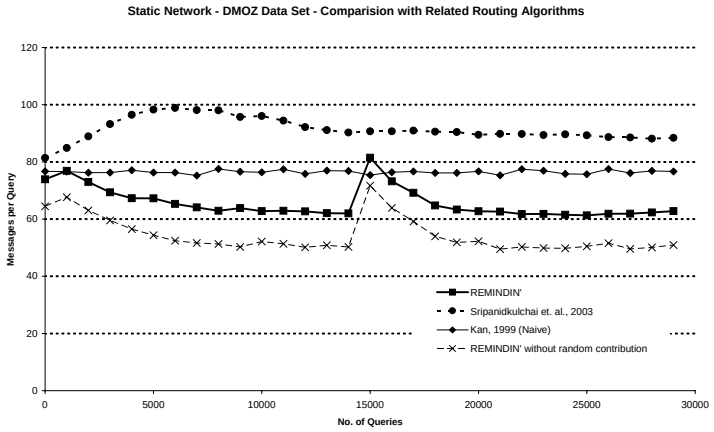


Figure 7.9.: Static Network: Comparison of Query Routing Algorithms: Messages per Query

more messages.

The more effective shortcut creation of REMINDIN' in comparison with IBL also influences the number of messages produced per query. Although the number of messages used by the algorithms is always lower than the theoretically maximum number of messages ($126 + 13 = 139$[7]), since the query messages are forwarded to the same remote peer through different routes and not processed further, the recommender layer enables many peers to learn quickly about the available knowledge in the network and a query is more often forward to the same remote peer. In the event of an interest shift the effect of the t_{greedy} parameter becomes obvious. In some cases no local shortcuts are similar enough to the new queries so that the algorithm falls back on the bootstrapping or the default network layer. As the selection of peers is more random in this cases the aforementioned query reduction principle does not apply any more and the number of messages increases.

Hypothesis 2 Figures 7.10, 7.11, 7.12 and 7.13 visualize the evaluation results of the comparison between REMINDIN', the naive approach and the IBL strategy in a dynamic network using the DMOZ data set. Peers issue queries for Web site links related to one topic in the DMOZ hierarchy. REMINDIN' and IBL use an index of size 40. REMINDIN' uses the similarity based peer selection strategy ($queryRelaxation = true$). During the simulation run each peer issues in average 30 queries approximately corresponding to a 2 hour period in a real file sharing network. After 15 queries the peers shift interest and choose queries from the second half of the available ones. Figure 7.8 focuses on the recall achieved by the different routing strategies. The total number of issued queries

[7]The number of hops is set to six, and two peers are selected each time. Thus the query can reach $2 + 4 + 8 + 16 + 32 + 64 = 126$ peers. On average 13 peers can provide a partial answer to a query.

is plotted against the achieved recall. The line 'Online available' indicates the maximal achievable recall as not all peers are always online. Likewise in the static network the recall of REMINDIN' increases with the number of sent queries and levels off at about 25% corresponding to 46% of the achievable recall. In the event of interest shift the recall nearly halves. It does not recover as strong as in the static case, though. The continues black line represents the mean of eight simulations runs with the same parameter setting. The gray range around it additionally visualizes the confidence interval for the simulation results.[8] The range shows that REMINDIN' is statistically significantly better than the related approaches. The IBL strategy produces recall levels of around 18% or 33% of the achievable recall with a fall in the event of an interest shift. The naive routing strategy results in circa 10% (19% of achievable) recall independently of the interest shift. Figure 7.11 plots the

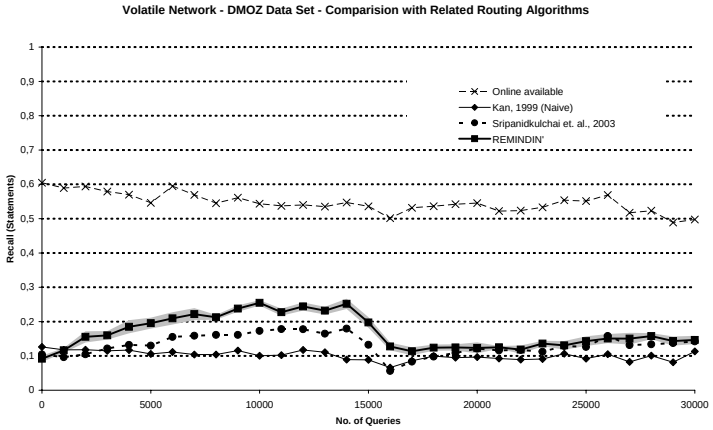


Figure 7.10.: Volatile Network: Comparison of Query Routing Algorithms: Recall

[8] Confidence Intervall The confidence interval of a mean is a range around the mean calculated from observed data (*cf. e.g.*, (Bol, 2004)). It expresses that the real mean of a data set falls with a certain probability – usually 95% ($p < 0,05$) – within this range. An observation is statistically significantly different from another observation if their confidence intervals do not overlap. The confidence interval for a mean is calculated multiplying the standard error (σ_n) of a mean with a t-value (Student's t-test). The standard error and the t-value depend on the number of repetition of an experiment.
Given a number of repetitions n of an experiment and observations for a data point x the mean μ, the standard deviation σ, and the standard error σ_n are defined as:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}; \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2; \sigma_n = \frac{\sigma}{\sqrt{n}}$$

The confidence interval of the mean is then

$$\mu \pm \sigma_n * t(p; n-1)$$

The experiments have been repeated $n = 8$ times such that $t(p; n-1) = t(0,05; 7) = 2,36$.

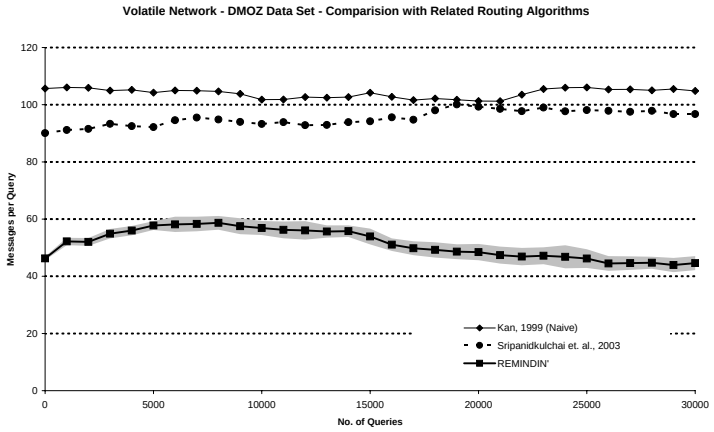


Figure 7.11.: Volatile Network: Comparison of Query Routing Algorithms: Messages per Query

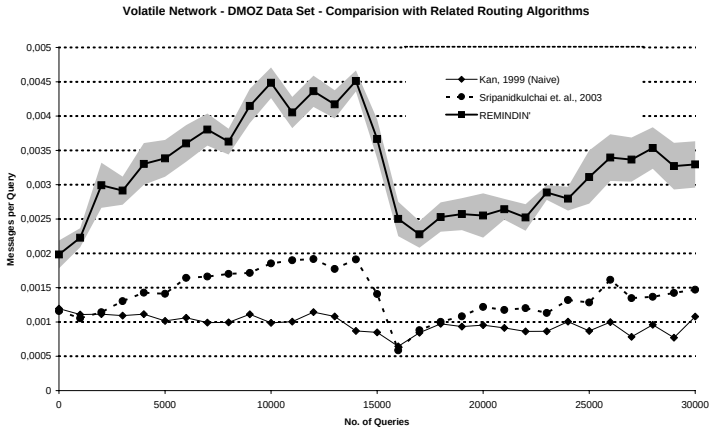


Figure 7.12.: Volatile Network: Comparison of Query Routing Algorithms: Message Gain

number of messages per query against the total number of sent queries. The number of messages per query produced by REMINDIN' decreases over time independently of the interest shift and levels off at around 45. The number of messages produced by the naive approach is constant at around 105. The number of messages produced by IBL increases a little bit and levels off at around 95 messages. The interest shift does not affect the number of messages produced by the IBL strategy.
Figure 7.12 combines the results of Figures 7.8 and 7.11 and plots the message gain against the number of sent queries.
Figure 7.13 plots the average number of hops required to find the first peer with an answer

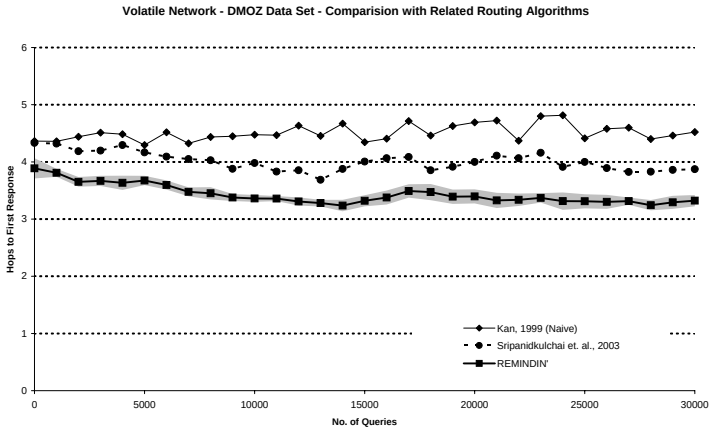


Figure 7.13.: Volatile Network: Comparison of Query Routing Algorithms: Time to First Response

against the number of sent queries. REMINDIN' finds knowledgeable peers quicker than the related approaches, while the IBL strategy is better than the naive strategy. The interest shift slightly increases this figure for REMINDIN' but it falls again afterwards.

The simulations validate the hypothesis that REMINDIN' has a better performance than related routing approaches also for dynamic unstructured P2P networks. The relative recall of REMINDIN' reaches almost the levels as observed in the static network, at least in the initial learning phase. After the interest shift these levels are not reached anymore. In a volatile network the bootstrapping peers correspond to the peers which are most of the time online. This behavior results in a decreasing number of messages per query over the entire simulation, but it also limits the discovery of new peers. The simulations related to Hypothesis 4 show that the omission of the bootstrapping layer increases the recall at the cost of more sent messages. The message gain figure is better with the bootstrapping layer activated.
IBL and the naive algorithm produce a different number of messages per query than in the static scenario, because the known remote peers on the network layer change with the simulation time. Peers which are online most of the time have a higher probability to end up on many network layers, reducing the variety in comparison with the static network.

Hypothesis 3 Figures 7.14 and 7.15 plot the evaluation results comparing REMINDIN', the naive approach and the IBL strategy in a dynamic network using the Bibster data set. Peers issue complex queries for bibliographic entries offering the BibTEX properties and a classification according to the ACM topic hierarchy. REMINDIN' and IBL use an index of size 40. REMINDIN' uses the similarity based peer selection strategy. During the simulation run each peer issues in average 30 queries approximately corresponding to a 2

hour period in a real file sharing network. Figures 7.16 and 7.17 use the same parameter setting with the synthetic data set and corresponding queries.

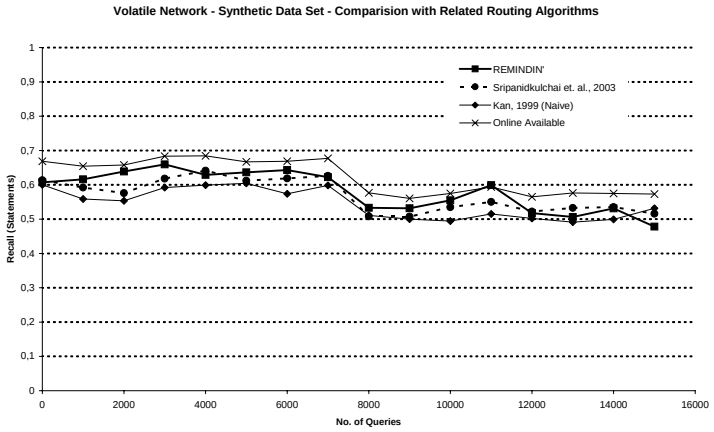


Figure 7.14.: Volatile Network: Comparison of Query Routing Algorithms – Conjunctive Queries – Bibster Data Set: Recall

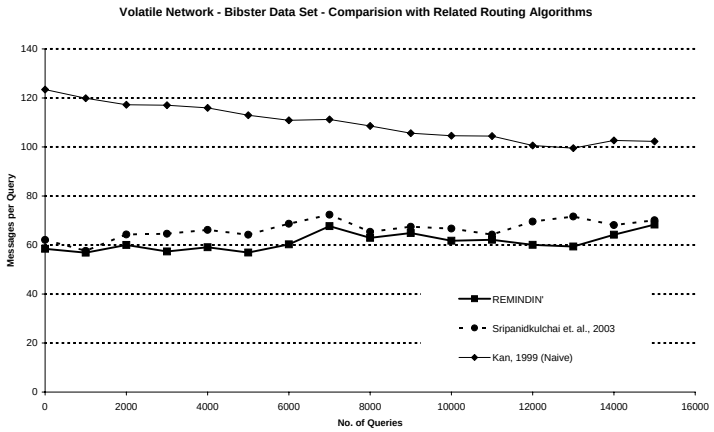


Figure 7.15.: Volatile Network: Comparison of Query Routing Algorithms – Conjunctive Queries – Bibster Data Set: Messages per Query

Figures 7.14 and 7.16 chart the recall for the three different routing approaches. 90% of the available answers are found independently of the used routing algorithm for the Bibster data set. The recall of REMINDIN' approaches 75% w.r.t. to the available content after 2000 queries and stays at this level during the simulation. The interest shift has a minor influence on the algorithms performance. IBL has also high recall levels at the beginning but its performance decreases in the course of the simulation and finishes with

60% recall w.r.t. the achievable recall. The recall of the naive algorithm is constantly 15% w.r.t. the achievable recall.

Figures 7.15 and 7.17 chart the messages per query for the three different approaches. REMINDIN' and IBL produce both in average 63 messages per query on the Bibster data set. The naive algorithm starts with 120 messages and finishes with 100 messages per query. On the synthetic data set REMINDIN' starts with 65 messages and produces 40 messages at the end of the simulation. The number of messages produced by IBL decreases

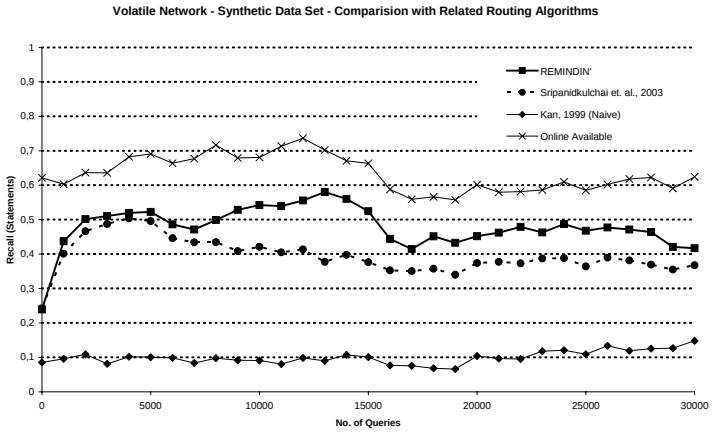


Figure 7.16.: Volatile Network: Comparison of Query Routing Algorithms – Conjunctive Queries – Synthetic Data Set: Recall

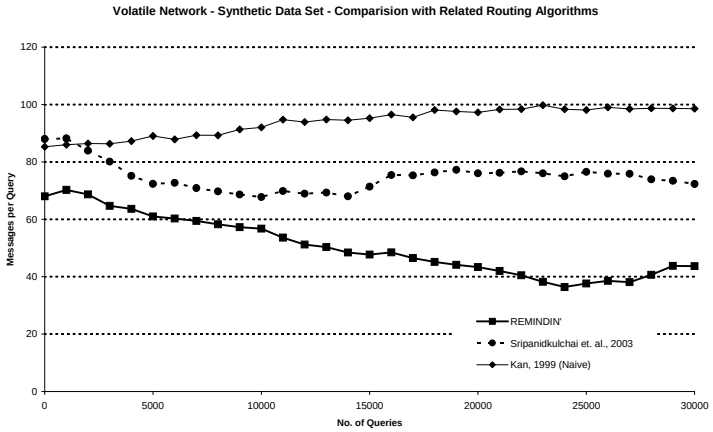


Figure 7.17.: Volatile Network: Comparison of Query Routing Algorithms – Conjunctive Queries – Synthetic Data Set: Messages per Query

from 90 to 70 in the course of the simulation.

The hypothesis that REMINDIN' is more efficient and has a higher quality of service in dynamic P2P networks using complex queries than related approaches can only be partially validated. On the Bibster data set all routing algorithm show the same performance levels. This is only partially conclusive, though. The distribution of the content is very skewed, therefore content location is particularly easy. The parameter setting for the maximum number of hops $maxTTL = 6$ is high for the small Bibster network. The bootstrapping layer does not show its message reducing effect, as most of the querying peers quickly create shortcuts. Only 398 peers did submit queries. On the synthetic data set the performance levels of the naive algorithm are comparable to the ones observed for the DMOZ data set. The difference between the IBL approach and REMINDIN' w.r.t. the recall figure is only gradual compared to the results of the single class queries. It is not intuitive that the total recall for both approaches is higher than for single class queries. The distribution of the data set explains this result. A smaller number of peers contribute more knowledge in the synthetic data set than in the DMOZ data set. Additionally learns REMINDIN' quicker, because it can create more than one short cut with each query.
The recall analysis of the IBL strategy shows that it decreases during the simulation time. This is due to its indexing strategy which is only based on time.
Chart 7.17 demonstrates that REMINDIN' reduces the number of messages sent in networks of the same size independently of the data set. IBL produces less messages for the synthetic data set than for the DMOZ data set, due to the concentration of content on fewer peers.

The interpretation of the evaluation results should consider, however, that the Bibster system is prototypical and the data set may not completely resemble observations obtainable from a commercial solution. Moreover, the synthetic data set is, though its properties are well founded, still a synthetic data set. Nevertheless, the results point in the same direction as the evaluations based on the DMOZ data set: REMINDIN' achieves the same or a higher recall than related approaches, while it may reduce the number of messages per query.

Hypothesis 4 Figure 7.18 illustrates the contribution of the different layers to the performance of REMINDIN'. Each chart is the result of using the indicated layer and all lower layers, *e.g.*, the content provider layer uses also the default network layer. The figure plots the message gain against the total number of sent queries. The simulations are setup with the same parameters as for Hypothesis 2.
The default network layer follows a naive routing strategy and offers a constant but low performance contribution. The use of content provider layer steadily increases the recall, while it leaves the number of messages per query unaffected. In the event of new queries the recall falls back and recovers only slowly. The recommender layer allows the algorithm to learn quicker and to achieve a higher recall. Finally, the bootstrapping layer reduces the number of messages per query, but slightly decreases the recall. In total, however, the message gain is at its highest level using all three layers.

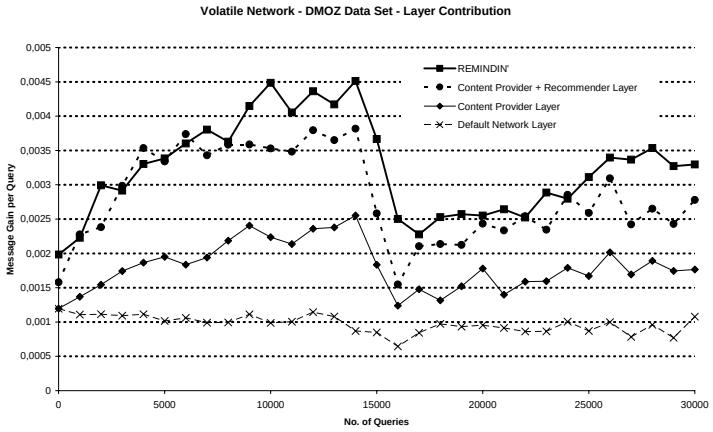


Figure 7.18.: Performance Contribution of each Overlay Layer

Hypothesis 4 is validated, because each layer contributes to REMINDIN' performance. The performance gains are achieved, because the content provider layer directs queries to knowledgable peers. The recall contribution of remote peers reached through content provider shortcuts is the highest. The recommender layer gathers information from queries which are routed through the local peer. Although this information is not precise it is a good indicator for the direction a query should travel. The direct recall contribution of remote peers reached through recommender links is low, but the selected peers forward queries to knowledgeable content providers. The bootstrapping metric favors remote peers which are well connected and receive a high number of queries. In case a peer has no information about any of the classes used in the query it will forward the query to the best bootstrapping peers. As the bootstrapping capabilities of the peers are disseminated in the network, more peers select the same remote peers and thus the number of messages decreases. The recall decreases slightly but less than the messages, thus the message gain increases. The application of the bootstrapping layer depends on the concrete application scenario. If the number of sent messages per query is crucial the bootstrapping layer offers an efficient way to reduce them. If an increase in messages is acceptable – from 45 to 85 in a network of size 1024 – the application of only the recommender layer is advisable.

Hypothesis 5 Figure 7.19 presents the evaluation results comparing the performance of REMINDIN' using different index sizes. The figure plots the message gain against the total number of sent queries. The simulations are setup with the same parameters as for Hypothesis 2.
The size of the index determines the required resource allocation for routing purposes at the local peer. The index size is constantly increased starting from 20, to 40, 100 and finally unlimited number of shortcuts per peer.

We observe that within a certain range the limit to the number of shortcuts stored at

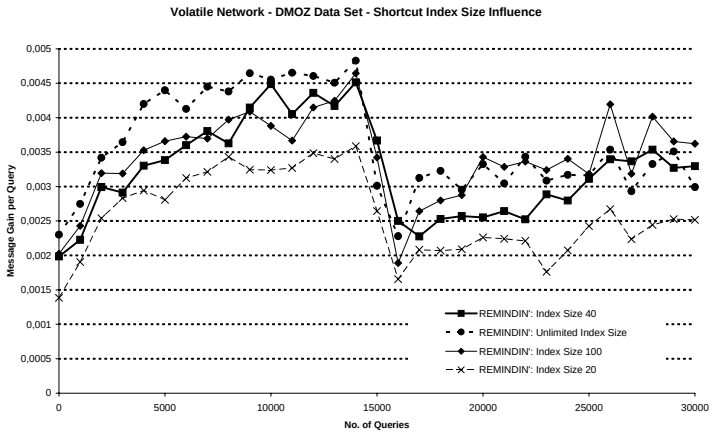


Figure 7.19.: Influence of the Shortcut Index Size

the local peer does not affect the results in terms of message gain. Peers store different shortcuts, because the ranking of the shortcuts partially depends on the local knowledge of the peer. Although the number of local shortcuts is low, the collective number of different shortcuts is high. This allows REMINDIN' to achieve the performance levels already with a relatively small index size. An index size between 40 and 100 is sufficient.

Hypothesis 6 Figures 7.20, 7.21, 7.22, and 7.23 visualize the evaluation results related to the effects of a t_{greedy} parameter variation. This parameter sets the minimal similarity between a query and a shortcut used for peer selection. If $t_{greedy} = 0$ all shortcuts are considered, otherwise shortcuts must be increasingly similar to the query in order to be considered. The simulations are setup for a dynamic network using the DMOZ data set. The DMOZ hierarchy is the reference ontology to determine the similarities between shortcuts and queries. The index size is set to 40. This evaluation compares additionally to the influence of the t_{greedy} parameter the two relaxation strategies ($queryRelaxation = true/false$).

Figure 7.20 plots the recall against the total number of issued queries. Independently of the value for t_{greedy} increases the recall with the number of issued queries, collapses in the event of an interest shift and slightly recovers afterwards. The simulations using parameter settings for t_{greedy} equal or below 0.5 perform distinctively better than simulations with parameter settings above 0.5. A decrease in the value for t_{greedy} increases the recall. The relaxation based peer selection mechanism ($queryRelaxation = true$) achieves a higher recall, faster than the similarity based peer selection mechanism with the same value for t_{greedy}.[9] A low value for t_{greedy} is in particular helpful in the event of interest shift, as the recall decrease is not as drastic as in cases with a high value for t_{greedy}.

[9]The bootstrapping layer is not activated for this strategy.

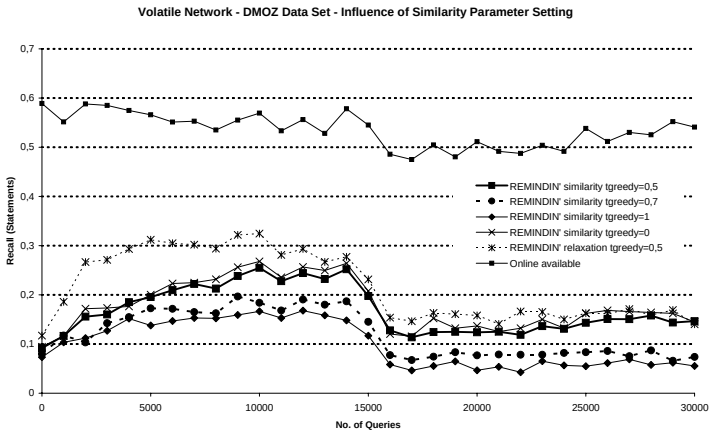


Figure 7.20.: Influence of Similarity Parameter Settings: Recall

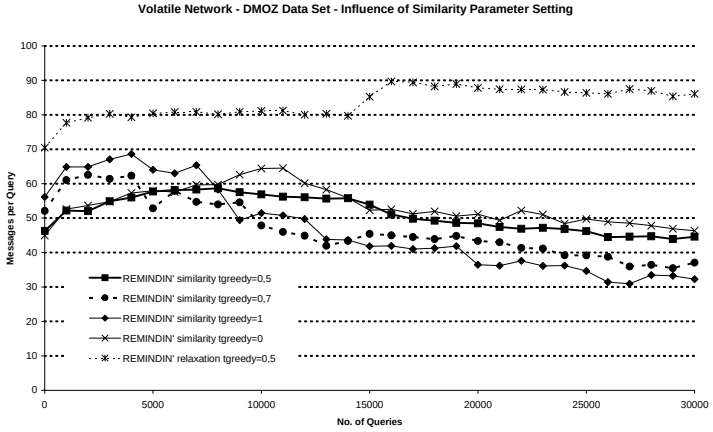


Figure 7.21.: Influence of Similarity Parameter Settings: Messages

Figure 7.21 charts the messages per query against the total number of sent queries. The number of messages per query is not as much influenced by the t_{greedy} parameter as the recall. In all cases using the similarity based relaxation strategy with an activated bootstrapping layer the number of messages per query falls with the total number of sent queries. A higher value of t_{greedy} coincides with a lower number of messages per query at the end of the simulation. The chart for the relaxation based strategy emphasizes the message reducing influence of the bootstrapping layer, as the number of messages per query is almost twice as high as in the simulations using the bootstrapping layer.

Figure 7.22 concentrates on the similarity based peer selection strategy and combines the recall and messages per query figures in the message gain.

Figure 7.23 emphasizes the tradeoff between the higher recall achieved with the relax-

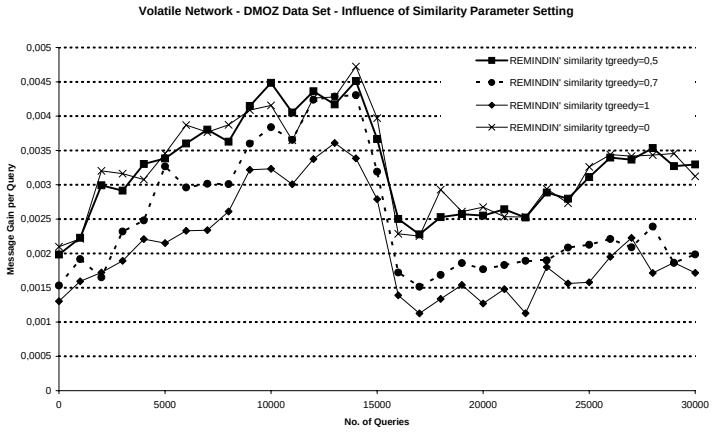


Figure 7.22.: Influence of Similarity Parameter Settings: Message Gain

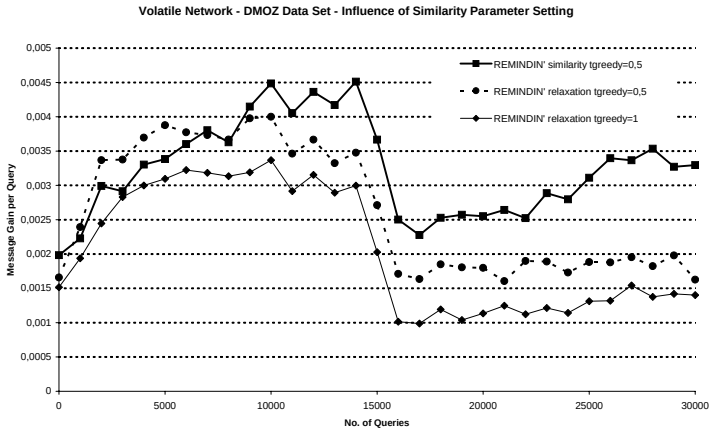


Figure 7.23.: Influence of Similarity Parameter Settings – Query Relaxation Approach: Message Gain

ation based peer selection strategy without the bootstrapping layer and the lower number message per query for the similarity based peer selection strategy using the bootstrapping layer. The highest message gain can be achieved with the similarity base peer selection strategy with the application of the bootstrapping layer.

The simulation results validate the hypothesis, that the use of semantics is beneficial for the peer selection process. The lower the value of t_{greedy} the more shortcuts are considered in the shortcut selection process. The algorithm favors shortcuts which are closer to the query. If peers locally store content which is semantically close to each other the service quality of the routing algorithm increases. This is particularly interesting in the

event of interest shift. The relaxation based peer selection offers for some cases a better performance than the similarity based peer selection strategy. It depends on the data set and the structure of the shared ontology which approach is better suited for an application. Similarity based approaches are better suited for ontologies in which classes are primarily connected through properties, because they can incorporate domain specific information into the similarity measure. Relaxation based approaches profit most from a deep class hierarchies, since the query is gradually broadened implying that the algorithm considers few additional peers in each relaxation step.

Hypothesis 7 Figures 7.24, 7.25 and 7.26 depict the influence of different parameter settings for the indexing strategy on the overall performance of REMINDIN'. The simulations are setup with the same parameter setting as for Hypothesis 2. In each simulation only the values set for $simInfluence, typeInfluence$, and $timeInfluence$ are changed. In REMINDIN' all three parameters influence the index management. The parameters are set to $simInfluence = 0.1, typeInfluence = 0.8$ and $timeInfluence = 0.1$. REMINDIN' LRU corresponds to the parameter setting in which only the time is considered to rank shortcuts for index maintenance ($simInfluence = 0, typeInfluence = 0, timeInfluence = 1$). Similarly REMINDIN' community considers only the shortcut type ($simInfluence = 0, typeInfluence = 1, timeInfluence = 0$). Finally, REMINDIN' similarity considers only the similarity of shortcuts to the local content ($simInfluence = 1, typeInfluence = 0, timeInfluence = 0$).

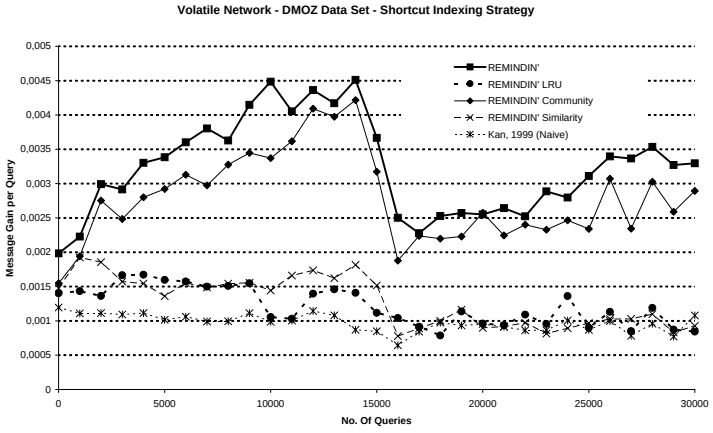


Figure 7.24.: Influence of Indexing Parameter Settings: Message Gain

Figure 7.24 plots the message gain against the total number of sent queries for four different parameter settings and the naive routing approach. The pure indexing strategies only based on similarity and time cannot increase the message gain levels above the levels observed for the naive algorithm. Only with the community strategy message gain levels comparable to REMINDIN' are achievable. No parameter setting alleviates the problem

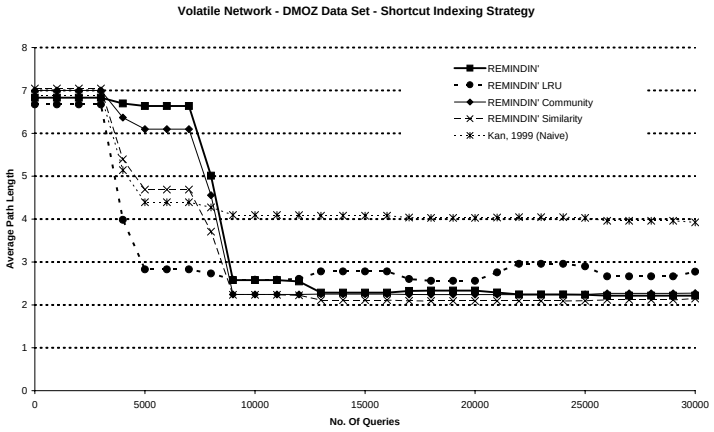


Figure 7.25.: Influence of Indexing Parameter Settings: Average Path Length

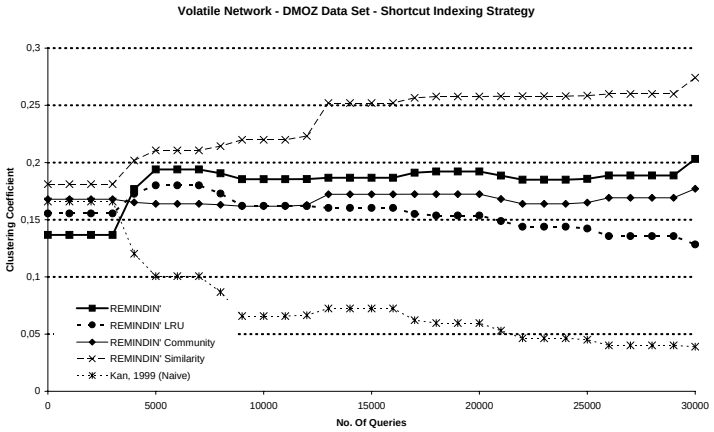


Figure 7.26.: Influence of Indexing Parameter Settings: Clustering Coefficient

that recall levels do not recover to levels observed during the learning phase. The bootstrapping peers receive a larger proportion of the queries, reducing the number of messages but also the recall as they are not aware of peers able to answer the new set of queries. As mentioned before with the exclusion of the bootstrapping layer from the query selection process this problem may be overcome. A user driven deletion of all indexing information is another option.

Figure 7.25 compares the development of the average path length for different routing approaches and parameter settings. Independently of the parameter setting the average path length decreases below the average path length observed for the naive routing approach during the simulation time.

Figure 7.26 compares the development of the clustering coefficient for different routing

approaches and parameter settings. The initial average path length and clustering coefficient are roughly the same for all settings, because the simulations start with a small-world network topology. The clustering coefficient of the naive algorithm decreases quickly after the simulation start. The clustering coefficient for REMINDIN' LRU and REMINDIN' community are unchanged during the simulation time. The clustering coefficient for REMINDIN' similarity increases steadily, while it increases slightly for REMINDIN'.

The naive algorithm establishes new connections to randomly chosen remote peers in the simulation, because some peers go offline and the initial network layer configuration changes. The resulting network topology is more similar to a random topology than to a small-world topology as indicated by the two measures.

The simulations support the hypothesis that different parameter settings for the indexing strategy influence the network organization. Increasing the influence of the $simInfluence$ parameter results in an increasing clustering coefficient. The REMINDIN' parameter setting combines the higher recall achievable through an emphasis on the $typeInfluence$ parameter and the higher clustering coefficient achievable through an emphasis on the $simInfluence$. In application scenarios in which the detection of persons with similar interest is more relevant than a high recall the parameter setting can be changed in favor for the $simInfluence$ parameter.

The recall for REMINDIN' LRU and REMINDIN' similarity are lower than for the combined strategy, because neither strategy considers the importance of content providers. Although peers with similar interests are clustered together the recall does not increase, because neither strategy distinguishes between content provider shortcuts and recommender shortcuts. Recommender shortcuts boost performance levels only in combination with content provider shortcuts.

7.5. Summary and Outlook

The evaluation of REMINDIN' on three different data sets demonstrates the applicability of this routing approach in pure unstructured P2P networks. REMINDIN' outperforms related routing approaches in terms of recall and messages in static networks. In dynamic networks REMINDIN' reaches the same recall levels as related approaches, but produces less messages per query. Further on it finds answers with less hops than related routing strategies. Depending on the application scenario its customization to use more messages to reach a higher recall is straightforward. The performance of REMINDIN' can be further fine-tuned by varying the parameter setting. A change in the shortcut index size slightly affects its performance. A variation in the parameter setting for the index maintenance allows to support the creation of clusters with semantically similar content.
The main finding of our evaluation studies is that the use of semantics is beneficial for routing. Due to its semantics awareness REMINDIN' achieves higher learning rates, adapts to changing user interests and clusters peers with similar interests better.

Nevertheless, the performance of REMINDIN' may be improved if we consider a number of open issues: The determination of the right parameter setting is laborious. For example the optimal number of hops ($maxTTL$) depends on the network size; a strategy for an automatic adaptation may be therefore beneficial. The same line of reasoning can be applied for the parameters related to the index maintenance, the t_{greedy} parameter or the bootstrapping threshold. REMINDIN' currently does not offer a solution for the case that no answers have been found. A combination with a feedback mechanism in which a query is first sent with a lower value for $maxTTL$ and only later if no results could be found with a higher value could be a solution for this problem. Such a strategy may also overcome the issue that the recall of the algorithm does not achieve higher levels after an interest shift. Finally, we mention the evaluation of different routing approaches using data sets from commercial solutions.

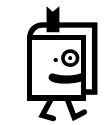
Part IV.

Related Work & Conclusions

"Das also war des Pudels Kern!"
— Johann Wolfgang von Goethe
Faust I, Vers 1323

8. Related Work

Und es gibt Dinge in denen auch der Geist ist.
— Anaxagoras



This part contains (i) **related work** and (ii) **conclusions**.

In **this chapter** we present related work on distributed knowledge management systems in Section 8.1, on ontology engineering methodologies and argumentation support in Section 8.2 and on query routing in Section 8.3.

8.1. Related Work on Distributed Knowledge Management Systems

The SWAP system was designed to meet the requirements of a DKM scenario. The following scenario properties influenced the system design to a great extent (*cf.* Section 3.2): The system supports the representation and sharing of user knowledge on their local machines. The knowledge as well as the peers are volatile. The system represents knowledge which is sensitive.
Comparing the SWAP system with related approaches to knowledge sharing we look in particular at these properties.

InfoQuilt The InfoQuilt system provides a framework for formulating complex information requests, involving multiple ontologies, and supporting a form of knowledge discovery (Arumugam et al., 2002). Likewise the SWAP system they use Semantic Web representation languages to formalize the exchanged knowledge. Their focus, though, is on efficient query processing as the system was originally designed to support queries on distributed databases. In their case the representation of user knowledge is of no concern, as they assume that queries should be executed on already available databases. Hence, they do not provide and do not need a methodology to support the construction of a shared ontology. As their system is designed to integrate a limited number of data sources they propose a mapping mechanism to enable integrated query processing. They do not address volatility of knowledge providers. In order to find relevant information sources and ontologies stored in the system they use centralized indices as opposed to our decentralized architecture.

EDUTELLA EDUTELLA is a P2P system based on the JXTA platform, which offers very similar base functionality as the SWAP system (Nejdl et al., 2002). The system is designed to support the exchange of learning material between educational institutions to facilitate its reuse. They have focused on the definition of a new meta query language to integrate existing query languages, such as SQL, by means of Semantic Web representation languages. The use case assumes a lightweight publishing process in which the learning material is annotated w.r.t. a predefined core ontology. Volatility of peers is of less concern in their use case and the published knowledge is publicly available. The have developed a routing algorithm which is further discussed in section 8.3.

EDAMOK EDAMOK is a DKM system also referred to as KEEx . Their use case is the same as ours (Bonifacio et al., 2002). The EDAMOK system thus provides functionality to integrate and share user knowledge; they do not use Semantic Web representation languages but their own formalism, though. The users in their scenario are also volatile and share sensitive information. Their research focus lies on the discovery of mappings between different user conceptualizations. Instead of offering a methodology to facilitate the building of shared conceptualizations they propose a technical solution to bridge between different conceptualizations. The application of their approach to knowledge integration in case studies, however, moved their research focus more in the direction taken by our methodology (Bonifacio et al., 2004b). They did not investigate routing strategies to find peers in their network as they relied on manual created links between the peers.

8.2. Related Work To DILIGENT

The related work discussion for the DILIGENT methodology is separated into the presentation of ontology engineering methodologies and argumentation frameworks supporting ontology engineering.

8.2.1. Related Ontology Engineering Methodologies

The development of the DILIGENT methodology is driven by the requirements, *i.e.*, decentralization, non expert builders, autonomy and iteration, as elaborated in Section 4.1.2, page 46. Describing the related work we will pay particular attention to these factors. Table 5.1, page 115 visualizes the differences between the major methodologies and DILIGENT.

The selection of related work is based on existing surveys on ontology engineering methodologies and recent publications. An extensive state-of-the-art overview of methodologies for ontology engineering can be found in Gómez-Pérez et al. (2003). Jones et al. (1998) compiled an early review on ontology engineering methodologies. More recently

Cristani & Cuel (2005) proposed a framework to compare ontology engineering methodologies and evaluated the established ones accordingly. A very practical oriented description for ontology building can be found in Noy & McGuinness (2001).

In our context, the following approaches are especially noteworthy.

IDEF5 IDEF5 is an ontology building methodology to support the creation of ontologies for centralized settings (Benjamin et al., 1994). 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 meets the requirements set in the beginning. Even though ontology evolution is supported by means of the extension of Proto-Concepts, decentralized development and issues around partial autonomy are not dealt with in IDEF5.

Enterprise Ontology The Enterprise Ontology 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 (Uschold & King, 1995; Uschold & Grüninger, 1996; Uschold et al., 1998b). The principles behind this methodology influenced many researchers in the ontology community, as they have been among the first to propose a solution for the ontology engineering problem. However, they do not provide methodological guidelines for the distributed and evolving ontology development.

CO_4 CO_4 (Cooperative construction of consensual knowledge bases) 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 (Euzenat, 1995; Euzenat, 1997). 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 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. CO_4 was evaluated in two case studies. It 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 as in DILIGENT.

METHONTOLOGY METHONTOLOGY is among the more comprehensive ontology engineering methodologies as it is one for building ontologies either from scratch, reusing other ontologies as they are, or by a process of re-engineering them (Gómez-Pérez, 1996; Fernández-López et al., 1999). The framework enables the construction of ontologies at the "knowledge level", *i.e.*, the conceptual level, as opposed to the implementation level. The framework consists of: identification of the ontology development process with the identification of the main activities, such as, evaluation, configuration, management, conceptualization, integration implementation; a life cycle based on evolving prototypes; and the methodology itself specifying the steps for performing the activities, the techniques used, the outcomes and their evaluation. They describe very detailed the process to build an ontology for centralized ontology based systems. They do not provide guidance for decentralized ontology development and do not focus on post development processes.

CommonKADS CommonKADS is not *per se* a methodology for ontology development (Schreiber et al., 1999). 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. CommonKADS, thus, does not support the creation of distributed knowledge management applications as they assume centralized development. Ontology evolution is not an issue in their methodology.

HOLSAPPLE & JOSHI In (Holsapple & Joshi, 2002) a methodology for collaborative ontology engineering is proposed. The aim of their work is to support the creation of a static ontology in a community process. 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 sent around, until all participants agree with the outcome. Their methodology neither supports the evolution of ontologies nor does it provide a structured model to capture the feedback from the domain experts.

On-To-Knowledge Methodology The On-To-Knowledge 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 (Sure, 2003). 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". 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 phases *Refinement - Evaluation - Application - Evolution* typically need to be performed in iterative cycles. In a nutshell the On-To-Knowledge 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 to systematically evolve an ontology neither does it provide support for structured argumentation.

HCOME In (Kotis & Vouros, 2005) the authors present a very recent approach to ontology development. HCOME stands for *H*uman-*C*entered *O*ntology Engineering *Me*thodology. 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 decisions based on the IBIS (Kunz & Rittel, 1970) model. After a discussion and agreement the ontology is moved to the Agreed space. Although HCOME supports the same use case as DILIGENT it focuses on the technical problems of it. HCOME does not provide a detailed description of the process steps to follow in order to reach agreement among the participants. Moreover, they have not extended the IBIS model which we found essential for applying it to ontology engineering discussions.

UPON The Unified Process for ONtology building (UPON), has been proposed in (De Nicola et al., 2005). 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 *Requirements identification*, *e.g.*, by writing a story board and using competency questions, *Analysis*, which includes the identification of existing resources and the modeling of the application scenario, *Design and conceptualization*, *Implementation* and finally *Test*, in which the coverage of the application domain should be guaranteed and the competency questions are evaluated. The work flows apply to the four phases *Inception*, *Elaboration*, *Construction* and *Transition* defined in the methodology. These phases are performed in a cyclic manner. After each cycle an applicable ontology is produced. Although UPON supports the evolution of ontologies it does not define methodological guidelines for decentralized ontology development. As it is recent work, it shows the raising interest in the research community in use cases treated by DILIGENT.

8.2.2. Related Argumentation Frameworks

The DILIGENT argumentation framework facilitates the building and evolution of shared ontologies in decentralized partially autonomous environments. The customization of the IBIS model for ontology engineering and its formalization are the two main distinguishing factors. Related work is discussed with a focus on these factors.

Easterbrook Easterbrook (1991) comprehensively summarizes the field of conflict mediation. He gives an introduction to economic and behavioral models to conceptualize and resolve conflicts in discussions and proposes a new model to systematically resolve conflicts. The proposed model starts with identification of the conflict type according to the model presented in (Shaw & Gaines, 1989). First, the parties establish correspondences between their different conceptualizations. Second, the identify conflicting issues. The conflicting issues are discussed supported by gIBIS. The parties externalize the assumptions behind the decisions and justify them. In this way goals and motivations become clear to all participants. For conflicting issues resolution criteria should be defined. In a next phase the participants should 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.
This work establishes the basic framework for argumentation based conflict resolution. The DILIGENT argumentation framework elaborates on this basis and introduces a framework focusing on ontology engineering discussions.

Skuce Skuce (1995) is among the first to mention the importance and the difficulties in reaching agreement on ontological commitments. His work focuses on the definition of an upper-level ontology. It is similar to our work as he proposes to start with an initial shared ontology, send it to the participants, collect comments and update it. He introduces an intermediate level between the natural language description of the purpose for an ontology and the formal model. In this intermediate level the assumptions, justifications and definitions should be captured in order to make assumptions underlying the formal model explicit. In contrast to DILIGENT the proposed model is only roughly described. It does not aim at the continuous evolution of the ontology in parallel to its usage, but at the creation of a static ontology with evolving prototypes. The discussion is not supported at a fine-grained level.

Compendium In (Buckingham Shum et al., 2002) a case study in engineering an ontology from the combination of three existing ones is described. The compendium tool (Selvin et al., 2001) is used to guide the discussion in a synchronous meeting.[1] The results of the case study show that structured argumentation is beneficial for ontology engineering. The traceability of the decisions was enhanced. However, the authors were more

[1] Compendium is a tool to support IBIS like discussions.

concerned with the evaluation of their tool than with the specific issues arising in an discussion concerning an ontology. The authors do not examine which kinds of arguments are exchanged and how the discussion could be made more efficient.

Aschoff *et. al.* Aschoff et al. (2004) 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 analyze the dominant types of arguments which are used in the discussion and neither provide a formal model to capture the argumentation.

KUABA The KUABA design rationale ontology is a formal model of the IBIS argumentation vocabulary (de Medeiros et al., 2005). 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. Neither does KUABA define a subset of arguments particulary suitable for software design nor do the authors report on a case study evaluation. KUABA demonstrates the interest in the topic in adjacent research communities.

8.3. Related Work on Routing in Peer-to-Peer Networks

Their exist a number of different approaches to routing in P2P networks. As discussed in Section 2.4.2.2, page 30 they can be categorized according to the two dimensions *centralization* and *structure*. The discussion of related routing algorithms follows this categorization. The selection of related work was influenced by the surveys (Milojicic et al., 2002), (Androutsellis-Theotokis & Spinellis, 2004), (Oram, 2001), (Tsoumakos & Roussopoulos, 2003b) and (Siebes, 2002).

REMINDIN' is a decentralized routing algorithm for unstructured P2P networks, thus meeting the autonomy and security requirements of our use case maintaining efficiently a high quality of service in a volatile application setting. REMINDIN' supports queries for semantically enriched content. We will review related work particularly considering these aspects.

8.3.1. Routing Algorithms for Centralized Peer-to-Peer Networks

GlOSS and CORI First approaches for efficient indexing in P2P architectures were central indices, that have to transmit metadata about the available content to central indexing peers, like *e.g.*, GlOSS (Gravano & García-Molina, 1995) and CORI (Callan et al., 1995). These systems were designed to support the selection of best fitting databases for a query. The algorithms establish non-forwarding indexing links between the indexed information sources and the central server. The user queries the central server, which in turn answers with the best fitting database which then answers the query. These systems, however, did not aim at very-large-scale, highly dynamic, self-organizing P2P environments, because they were not an issue at the time these systems were developed.

Napster *Napster* Shirky (2001) was one of the first applications which raised public interest in the P2P paradigm. Napster facilitated the exchange of music files between peers. Peers uploaded their music file descriptions to Napster servers and users could search it. The results contained the peer identifiers where the proper music file could be downloaded. Napster was closed down due to copyright problems on the shared data.

As discussed in the Section 6.1.3 a centralized system does not meet our requirements.

8.3.2. Routing Algorithms for Super-Peer-Based Peer-to-Peer Networks

EDUTELLA EDUTELLA uses a super-peer-based routing mechanism based (Nejdl et al., 2003). Peers which have topics in common are arranged in a hypercube topology. This topology guarantees that each node is queried exactly once for each query. It is thus a very efficient approach to flood queries to all online peers. From the requirements perspective this solution for query routing is not adequate in our use case as the peers have to publish their expertise to the super-peers. Their are also technical problems in highly volatile P2P environments, as the hypercube topology must be maintained. Our algorithm is not based on an explicit topology, thus it does not generate any overhead to establish it. Our simulations illustrate that we need much less messages per query than the number of peers available in the network in order to reach the most knowledgeable ones. REMINDIN'cannot ensure a complete answer, though.

A second routing algorithm for the EDUTELLA system uses content provider shortcuts for document retrieval (Balke et al., 2005). Only exact matches between content provider shortcuts and queries are considered for peer selection. The peers publish their local indices in a static super-peer network. The shortcut index policy considers temporal locality, each index entry has a certain time to live after which the shortcut has to be reestablished for the next query on that topic. In contrast to REMINDIN' they only consider exact matches between shortcuts and queries, hence they do not investigate complex ranking metrics for semantically similar shortcuts. They do not use the concept of recommender peers.

KAZAA We selected KAZAA as an example for currently running popular P2P file exchanges (Sharman Networks, 2006). KAZAA uses super-peers to locate content in the network. KAZAA and other public P2P systems do not offer extended semantic querying capabilities, such as the ones required in our scenarios, instead search is restricted to a limited number of attributes, *e.g.*, a file name, or a song title.

8.3.3. Related Routing Algorithms for Decentralized Peer-to-Peer Networks

The discussion of related routing algorithms for decentralized P2P networks is divided into the review of structured and unstructured approaches to routing.

8.3.3.1. Routing Algorithms for Structured Decentralized Peer-to-Peer Networks

DHT One of today's main technique for indexing P2P systems are so-called distributed hash tables (DHTs). As discussed in Section 2.4.2.2, page 30 they come in two flavors. In the one approach a hash key is generated from the indexing terms and the mapping between keys and peers is stored in a well defined way. In the other approach the content itself is replicated in a well defined manner in the network. We survey some of the more prominent implementations of such networks. All of them could not be applied in our use case as they do not support queries for semantic content. Furthermore, recent research shows that due to the publishing/unpublishing overhead, DHTs lack efficiency when highly replicated items are requested (Loo et al., 2004). The performance decreases further if the network has to cope with a high churn rate. Content on the desktop of users is co-located with other relevant items and can thus be used in a DKM setting. A disadvantage of structured P2P topologies for distributed knowledge management is that structured overlays destroy this locality meaning that enhanced opportunities for browsing are lost (Keleher et al., 2002).

Structured P2P topologies are different w.r.t. three efficiency criteria: the number of hops a request has to travel until the key is found, the size of the index at each peer and the number of peers which are contacted when a peer leaves or joins the network. The *Chord* algorithm organizes keys or documents in a ring (Stoica et al., 2001). The identifier space is uni dimensional. In a Chord ring a key can be found in $log(N)$ hops, while each peer maintains a list of $log(N)$ entries, with N the number of peers connected to the network. Joining the network causes $log(N)^2$ messages. The identifier space in a *CAN* topology is multidimensional (Ratnasamy et al., 2001). The number of hops to retrieve a key may be reduced by increasing the number of dimensions, while the size of the index and the number of insertion operations increases with number of dimensions. *Pastry* organizes the keys in a Plaxton style global mesh (Rowstron & Druschel, 2001). In contrast to Chord only $log(N)$ peers need to be notified if a peer joins or leaves the network.

pSearch pSearch builds on the CAN model and combines information retrieval techniques with CAN's key generation function (Tang et al., 2003b; Tang et al., 2003a). The

system is designed to support keyword based document retrieval. They use the vector space model and latent semantic indexing in order to generate keys which place the corresponding content in the network according to its 'semantic' similarity. The semantic similarity is determined by the latent semantic indexing function. This approach to structure topologies facilitates the answering of requests with more than one query term, while maintaining the efficiency of DHT. It remains restricted to keyword based search, though.

P-Grid Gridvine is build on top of the P-Grid P2P infrastructure (Aberer et al., 2004; Aberer et al., 2003). P-Grid offers the classical properties of DHT building on a balanced tree and order preserving hashing functions. The Gridvine extension allows for querying RDF(S) structures stored in the P-Grid system. Gridvine stores RDF triples in the DHT. A peer is responsible for all triples related to a specific resource. Besides the storing and retrieval functionality for RDF it integrates a semantic gossiping algorithm which allows for the detection of correspondences between different schemas stored in the network. Gridvine is not applicable in our scenario as triples are stored at well defined peers in the network which are not related to the peer creating it.

8.3.3.2. Routing Algorithms for Unstructured Decentralized Peer-to-Peer Networks

For application scenarios where DHT approaches to routing are not adequate, such as ours, different routing algorithms have been proposed.[2]

Gnutella The Gnutella protocol is a very early routing algorithm for unstructured P2P networks (Kan, 2001). In Gnutella remote peers are detected by ping messages within the local IP subnetmask. The detected remote peers build the neighborhood of a Gnutella peer with an average size of 5. A request is routed in the network by flooding it, *i.e.*, by sending it to all neighbors. Answers are returned directly to the querying peer. A request is tagged with a maximum number of hops (TTL) and contains an ID in order to prevent that one peer answers a query twice. As the flooding approach to query routing does not scale to large number of peers for obvious reasons, the random walk algorithm has been proposed by Lv et al. (2002). A predefined number of random walkers is originated at the querying peer and travels the network. At each remote peer the random walker checks with the originating peer, whether an answer has been found and whether the walker should continue or not. The random walker query routing is from a performance point of view comparable to the naive approach presented in this thesis. We compared REMINDIN' to the naive approach to routing and show REMINDIN's superior performance.

Small-World The small-world characteristics of P2P networks and their emergence are the topic of numerous research contributions (*cf. e.g.*, (Watts & Strogatz, 1998; Faloutsos

[2] See Section 6.1.3 for a discussion why DHT's are not suitable for our scenario.

et al., 1999; Kleinberg, 2000; Pujol et al., 2004)). Search strategies exploiting small-world characteristics are scrutinized in, *e.g.*, Adamic et al. (2001). Two ideas are proposed: A peer establishes search links with remote peers of a high out-degree. Applying a flooding strategy in such a network reaches a large portion of the available remote peers with a relative small number of hops[3]. This strategy produces many messages ensuring a high quality of service. According to the second idea non-forwarding indexing and forwarding search links are established between the local peer and a remote peer with a high in-degree. A request is sent to a remote peer which is aware of the content of many other remote peers. It can thus forward the request to the most promising remote peer.
A routing strategy according to the second idea does not meet the requirements of our use case, as peers must advertise their expertise. The first idea influenced the definition of the bootstrapping layer of REMINDIN'.

SON In order to improve Gnutella's performance Crespo & Garcia-Molina (2002a), Crespo & Garcia-Molina (2002b) introduce two new query routing solutions for keyword based document retrieval. They explore the advantages of locally storing the query results from different peers in combination with a number of indexing terms from the retrieved documents (Crespo & Garcia-Molina, 2002a). This approach is comparable to the content provider layer in REMINDIN'. They later introduce the notion of semantic overlay layers (Crespo & Garcia-Molina, 2002b). Based on the clustering results of local documents a peer joins different semantic overlay layers each responsible for a specific cluster. The content is categorized according to shared topic hierarchies. In this algorithm super-peers are used to decide upon the assignment of peers to clusters. A request is handled by the overlay to which this document presumably belongs. While the specific technique is only applicable to documents they do not mention how to manage the overlay index in a completely distributed scenario. Likewise our approach they make use of the shared topic hierarchy to identify promising peers if no exact matches for a query can be found.

Anthill Anthill is a framework to support different P2P applications, such as file sharing and distributed computing (Babaoğlu et al., 2002). In the Anthill framework a peer is associated with a nest of ants, while ants represent messages. An ant can travel from one nest to another to fulfill its task. For the file sharing scenario each nest stores a number of keyword-hash – nest pairs. An ant can evaluate the list and decide to which nest to move next. If an ant has found an answer to a request it travels back its route and updates the keyword-hash – nest pairs in the visited nests. Although they have applied this approach to routing only for keyword based search, it is comparable to our creation strategy for recommender shortcuts. In our scenario we cannot inform the peers on the message path about the result of the query, as this would contradict our security requirements. They do not exploit any semantic relaxation techniques to enhance the peer selection quality.

[3] In Miligram's original experiments 5 to 6 hops were sufficient to connect any American with each other. Today, this average degree of separation, has decreased to 4 to 5.

Fireworks model The query routing algorithm presented in Hang Ng et al. (2003) considers first the creation of clusters of semantically close peers and second the exploitation of these clusters through a fireworks routing strategy. They define a general similarity function which can be based on arbitrary attributes. Peers publish the values of these attributes in the network, while receiving peers decide based on the similarity whether they keep search links to the remote peer or not. Clusters of semantically close peers emerge. A query is first sent on a random walk into the network. If the query encounters a peer which can answer the query the query is forwarded to all neighbors of that remote peer. This query routing algorithm can be adapted to meet the requirements of our scenarios. Their paper, however, remains too general to be applied directly to our use cases. We have tested the approach with our similarity function, but realized that we could not substantiate significant performance gains with our data sets, because most queries target at information stored at few peers only.

PlanetP PlanetP concentrates on P2P communities in unstructured P2P networks for keyword based document retrieval (Cuenca-Acuna et al., 2002). They introduce two data structures for searching and ranking which create a replicated global index using gossiping algorithms. Each peer maintains an inverted index of its documents and spreads the term-to-peer index. Thus forwarding indexing links are created. Inspired on the simple TF-IDF metric and based on this replicated index a simple ranking algorithm based on the inverse peer frequency is implemented. They focus on the reduction of advertisement messages sent when a peer joins the network. They only resend an advertisement when the content has changed significantly and the retrieved answers do not correspond to the index information. Advertisement based approaches do not meet the requirements of one of our scenarios. For the Bibster use case, however, the advanced advertisement strategies can be considered.

Interest based locality (IBL) Sripanidkulchai et al. (2003) introduce a routing algorithm exploiting interest-based locality in a static network. The system is used for keyword based document retrieval. They create shortcuts to remote peers which have successfully answered a query. These shortcuts are later exploited to forward requests. They are thus comparable to content provider shortcuts or non-forwarding search links. A new request is first sent to all peers listed in the shortcut index, but the remote peers do not forward the query. If the first strategy does not return any results, the query is forwarded according to the Gnutella protocol. To update the index they use a LRU strategy. In comparison to REMINDIN' they do not exploit semantic similarity between queries and shortcuts to select peers neither do they use shortcuts for forwarding. Our simulations show that REMINDIN' is better suited for the tested scenarios than the IBL approach.

Different indexing strategies for the IBL approach are investigated in Voulgaris et al. (2004), which influenced the design of our own indexing strategy. They find that an indexing strategy considering the number of retrieved documents outperforms a pure LRU

strategy. In our indexing strategy we additionally consider semantic similarity which particularly assists community creation.

Intelligent Search The Intelligent Search (Kalogeraki et al., 2002) and the Adaptive Probabilistic Search (Tsoumakos & Roussopoulos, 2003a) algorithms are both designed to optimize keyword based search for documents. The use local index information obtained from queries and answers from remote peers to guide a query through the network. This is comparable to our content provider shortcuts. Additionally they inform all peers on the message path about the result of the query. This is comparable to our recommender shortcuts. They compare their algorithm in an information retrieval scenario with the Gnutella protocol and show the algorithm superior performance. They do, however, provide active feedback to the peers on the message path about the quality of the search result. This does not meet the requirements for our application, as a peer may only return an answer to the querying peer does not want to inform others about his expertise.

Bibster For the Bibster system two routing strategies are implemented. In Haase et al. (2004b) an expertise based peer selection algorithm is introduced. From the local content the expertise of a peer is derived and represented in a shared topic hierarchy. The expertise is advertised to remote peers found on the network layer through non-forwarding indexing links. A peer decides based on the semantic similarity between the advertisement and its local expertise whether to store or discard the advertisement. A semantic topology emerges. Drawbacks of this approach to routing are that the decision if a peer knows about a specific topic is a yes / no decision which does not take into account the quantity of expertise and that the indexing links are static. The publishing of expertise produces some message overhead when a peer joins the network.
As the algorithm requires the advertisement of expertise it does not completely meet the requirements of the IBIT scenario. Besides we show in Tempich & Staab (2005) that REMINDIN' provides a higher quality of service after a warm-up phase as REMINDIN' adapts dynamically with new queries. We also investigate the combination of the two routing algorithms with the result that the combined algorithm learns faster, is less affected by changing queries and reaches generally a higher recall.

Infobeacons The Infobeacons system supports the selection of databases from keyword queries (Cooper, 2004). An Infobeacon is assigned to each participating database in the distributed system. An Infobeacon stores for successful queries important keywords and query terms of the retrieved documents with the number of retrieved documents. This is comparable to the content provider shortcuts we introduce for RDF resources. We have adapted the Infobeacons ranking strategy for conjunctive queries. As the system is designed for document retrieval it does not use semantics for query relaxation or as an input for an indexing strategy.

More recently they have studied load balancing strategies on unstructured P2P networks by comparing different approaches to create and maintain search and indexing links (Cooper & Garcia-Molina, 2005). They conclude that efficient load balancing is possible using only local information. They do not use semantic information for the decision whether a search link or an indexing link should be created, though. Their analysis is based on different parameters for a random based selection, while they consider a semantically motivated decision procedure as future work.

8.4. Summary

The review of the related work emphasizes the contribution of this thesis.

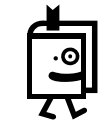
A number of research prototypes have been implemented to support knowledge sharing with P2P systems. The SWAP system was the first which was customized for distributed knowledge management settings and enabled the exchange of knowledge represented in terms of RDF(S).

Research in ontology engineering methodologies is mature. A number of elaborated ontology engineering methodologies exists. They support the creation of ontologies for centralized ontology-based applications. Currently a number of methodologies addressing the requirements of decentralized ontology creation and evolution are under investigation. DILIGENT is the first reporting on case study experiences and providing a detailed description of the process.

The support of knowledge acquisition with an argumentation model has been investigated before. However, empirical findings show, that existing argumentation methods are too general, for the ontology engineering task. DILIGENT proposes a customization of conventional argumentation models, which is adequate for the consensus building process in ontology engineering discussions.

Routing in P2P systems has attracted many research contributions. Most of the existing solutions are not applicable to our scenario for functional or security reasons. REMINDIN' is unique in this field as it has higher performance levels than related routing approaches and supports explicit semantics.

9. Conclusions


Overview

In **this chapter** we summarize the main contributions of this work in Section 9.1 and present an outlook to future work and research directions in Section 9.2.

9.1. Summary

Knowledge workers in contemporary organizations increasingly work in decentralized autonomous teams in order to cope with the rapidly changing business requirements. The underlying paradigm shift from the centralized hierarchical organization towards the decentralized autonomous organization of knowledge workers was early predicted by Peter F. Drucker in his study on the influence of information technology on the society. In order to adequately support knowledge workers in collecting, sharing and deploying knowledge from a technical point of view distributed knowledge management systems have been proposed. Apart from the direct analogy between decentralized autonomous teams and nodes in distributed knowledge management systems, these systems promise cost advantages and lower barriers of entry to knowledge sharing in comparison to centrally organized knowledge management systems. However, the decentralized system organization encourages heterogeneous knowledge structuring, makes knowledge location difficult and raises security concerns.

This thesis analyzes the requirements arising in distributed knowledge management systems and has three main contributions.

1. It proposes a system architecture and implementation for a distributed knowledge management system.

2. It proposes an ontology engineering methodology supporting the consensus building process in distributed, autonomous and evolving ontology engineering settings.

3. It proposes a novel routing algorithm enabling knowledge location in distributed knowledge management systems.

The SWAP system is a P2P infrastructure supporting distributed knowledge management. Knowledge representation in terms of the Semantic Web language RDF enables

knowledge exchange between peers in the network. The SWAP system supports integration of local knowledge sources into a local node repository and their annotation with semantic information. It offers visualization, retrieval, and integration services to facilitate knowledge sharing and enables the location of people with a certain expertise in the organization. The SWAP system is customized to two uses cases: XAROP supports distributed knowledge management in the tourism domain; Bibster supports the exchange of bibliographic information between researchers. The system has been evaluated in two case studies.

DILIGENT is an ontology engineering methodology addressing the requirements of the distributed knowledge management scenario. A board comprising ontology engineers starts the process by building a small initial ontology which is distributed to the users. The users are allowed to locally adapt the shared ontology in order to comply with changing business requirements. The user changes serve as input for a next version of the shared ontology. A board of ontology engineers and users updates the shared ontology in the central analysis and revision stage. The users locally update their shared ontology to the new version. In this way the shared ontology responds to emerging requirements, while the process allows for cost savings through small set up costs in comparison to a central approach.

A major issue in DILIGENT relates to the consensus building process, because the heterogeneous knowledge models created by the users should be partly integrated into the shared ontology. DILIGENT supports the consensus building process extending an existing argumentation model and adapts it to the requirements of ontology engineering discussions. It suggests a restricted set of argument types, thereby offers a systematic guidance for the discussions. As a result the agreement process becomes more structured and traceable. The methodology has been successfully evaluated in three case studies. The requirements on ontology engineering of the distributed knowledge management scenario are similar to the ones of the Semantic Web. The process is thus promising for this larger scenario, too.

REMINDIN' is a novel routing algorithm enabling knowledge location in pure unstructured peer-to-peer networks. REMINDIN' uses only local knowledge to direct complex queries to knowledgeable remote peers in the network. It uses responses to queries as well as information from queries routed through the peer in order to build up a local shortcut index. It uses semantic information of the ontology for peer selection and index maintenance. REMINDIN' is efficient in terms of resource consumption and effective in terms of peer selection. REMINDIN' follows a layered approach to query routing, which offers the possibility to customize it to scenarios in which a high recall is more important than a low number of messages per query or vice versa. Evaluations on three data sets demonstrate that REMINDIN' has higher performance levels w.r.t to recall and messages per query in static and dynamic networks than related routing approaches. Furthermore, the simulations show that the use of semantics is beneficial for query routing. On the one hand explicit semantics can help to cope with interests shifts in the query behavior of peers. On the other hand it enables the clustering of peers with similar interests. REMINDIN' thus

supports socializing in distributed knowledge management scenarios.

This thesis provides solutions for some of the organizational and technical challenges encountered in distributed knowledge management. Accounting for the empirical findings of this work in the next section we point out a number of future research directions for improving the economic viability of current approaches to distributed knowledge management.

9.2. Outlook

The SWAP system provides an infrastructure to solve many of the technical challenges in a distributed knowledge management environment. In order to become a commercially viable solution it should be none obtrusively integrated into the work environment of its potential users, *e.g.*, on the desktop. Current developments in the field of desktop search engines point in this direction. The integration of data from multiple applications is a second requirement for an eventual economic break through. The results achieved in research related to the social semantic desktop may contribute to the alleviation of this problem.

However, a serious thread to any further development of commercial P2P system is US Supreme Court ruling from the 27th of June 2005. It holds P2P application providers responsible for any copyright infringements supported by the platform, such as downloading music files. A byproduct of a knowledge exchange platform is that it also supports the exchange of copyright protected information. Future research should find solutions enabling knowledge exchange while maintaining the copyrights of the owners.

DILIGENT has demonstrated its use for the development of ontologies in distributed knowledge management settings. The next challenges come with its application in the larger Semantic Web context. The process may be particularly interesting for projects concerned with large scale ontology development, such as currently ongoing in the life science domain. For this to happen, the supporting tools should meet the usability requirements of these communities. Current developments related to the Web 2.0 could provide the infrastructure to build lightweight ontology engineering platforms supporting the DILIGENT methodology.
With an increasing deployment of ontologies in commercial applications cost estimations related to the creation and maintenance of ontologies become more important. As DILIGENT covers the entire life cycle of the ontology development it can also be the basis to calculate the total cost of ownership related to an ontology.
Another open issue in ontology engineering methodologies relates to the advances in the field of ontology learning. Although these techniques are not yet perfect, they offer acceptable performance levels to support the ontology engineering effort. However, only an integration into the overall ontology engineering process ensures that these performance gains can be lifted in ontology development projects.

As demonstrated with the help of simulations REMINDIN' improves the routing quality

in pure unstructured P2P networks. Future research could analyze routing approaches for unstructured P2P networks analytically to determine maximal performance levels. The combination of structured and unstructured approaches to routing is also promising. The network layer could be organized in a structured overlay while the semantic layers remain unstructured. The advantage of such an approach is that broadcasts can be executed more efficiently. Another option is the combination of advertisement based strategies with the passive strategy of REMINDIN'. This may be feasible for knowledge areas were privacy is of less concern. In any case the community could profit from more standardized test data sets in order to make the evaluation results of different groups comparable.

The starting point of this work is the postulate that distributed knowledge management systems offers advantages in comparison to centralized approaches in that it causes lower setup costs, lowers the barrier of entry for knowledge sharing and better fits to the knowledge creation process in organizations. The decentralized autonomous knowledge management organization comes with a number of organizational and technical challenges. This thesis proposes some solutions on the organizational as well as on the technical side to cope with these challenges and underlines the feasibility of the distributed approach to knowledge management.

Part V.

Appendix

‘Synonyms, ordered by estimated frequency’ search for noun ‘appendix’:

Sense 1
appendix *– (supplementary material that is collected and appended at the back of a book)*
=> *addendum, supplement, postscript – (textual matter that is added onto a publication; usually at the end)*

Sense 2
appendix, vermiform appendix, vermiform process, cecal appendage *– (a vestigial process that extends from the lower end of the cecum and that resembles a small pouch)*
=> *process, outgrowth, appendage – (a natural prolongation or projection from a part of an organism either animal or plant; ‘a bony process’)*

A. Evaluating the SWAP Metadata Model

Section 3.4.1 introduces the SWAP metadata model. The model captures provenance and other information required by the SWAP system and the routing algorithm. This section evaluates the efficiency of this approach to store metadata.

A.1. Evaluation Methodology

In order to evaluate the efficiency of this approach to store metadata we analyze three major use cases:

Use Case 1 The peer adds new content to the local node repository. He uses OntoScrape to extract local information from his machine and integrates it with the existing knowledge in the local node repository. In this case the Metadata Integrator adds provenance information according to the SWAP metadata model to the extracted information before it is integrated with the existing local knowledge.

Use Case 2 The peer uses the information stored in the metadata model in order to select remote peers in case a query is routed through the peer or it sends a query itself. In this case the peer retrieves metadata information related to the resources occurring in the query.

Use Case 3 The peer ranks all available remote peers according to their bootstrapping value. It retrieves all peer objects from the local node repository.

The efficiency is determined in terms of the required time to accomplish the tasks related to a use case. For evaluation purposes a data set generated from the DBLP web site[1] is used. The DBLP data set includes nearly 420.000 statements describing 126.000 instances. Each instance is defined by an average of three statements, which classify it according to the ontology, state the title and its type.

A.2. Evaluation Results

All tests were performed on a 1 GHz machine with 512 Mb RAM. In the figures the abbreviations have the following meaning:

[1] http://www.informatik.uni-trier.de/~ley/db/

$$
\begin{aligned}
S_{statement\ type} &= \text{statement} \\
|S| &= \text{number of statements} \\
SO &= \text{swabbi-object} \\
|SO| &= \text{number of swabbi-objects} \\
SP &= \text{swabbi peer-object} \\
|SP| &= \text{number of swabbi peer-objects}
\end{aligned}
$$

Use Case 1 The local node repository is empty and all statements are added the first time. The test is performed with an in-memory version of the LNR. The time consumed including all available DBLP data in the repository is compared with the time consumed by the metadata integrator.

Figures A.1 and A.2[2] depict the time used to integrate new statements. The experiments are repeated with a varying number of statements to integrate. While Figure A.1 concentrates on the content integration from the local peer, Figure A.2 visualizes the effects of integrating knowledge from remote peers into the LNR. Generating the new statements takes about as much time as integrating the statements into the repository. The annotation of the content statements doubles the integration time. The complexity is $O(|S| * |P|)$.

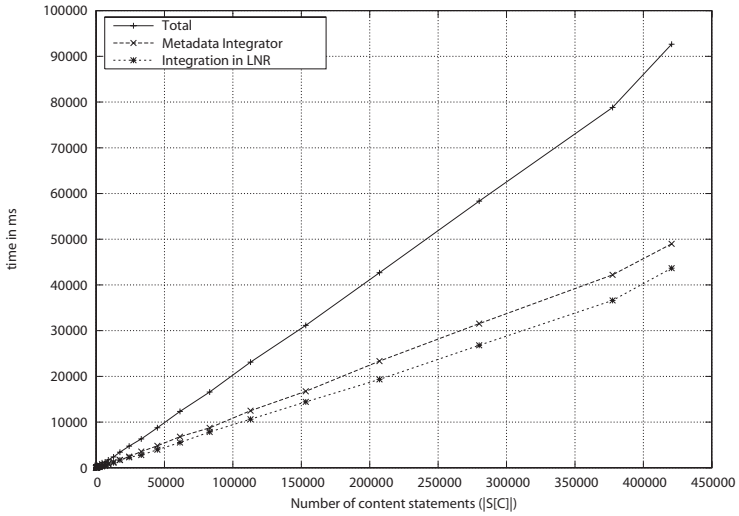


Figure A.1.: First Time Storage Annotation Times

Use Case 2 In this case we observe the time consumed to retrieve a swabbi-object from the LNR. Figure A.3 plots the time to retrieve one swabbi-object against the number of content statements. The time represents the average of ten runs. The figure

[2]Note that 1.000ms = 1s; 100.000ms = 1'40"min; 10^6ms = 16'40"min; 10^7ms = 2:46'40"std.

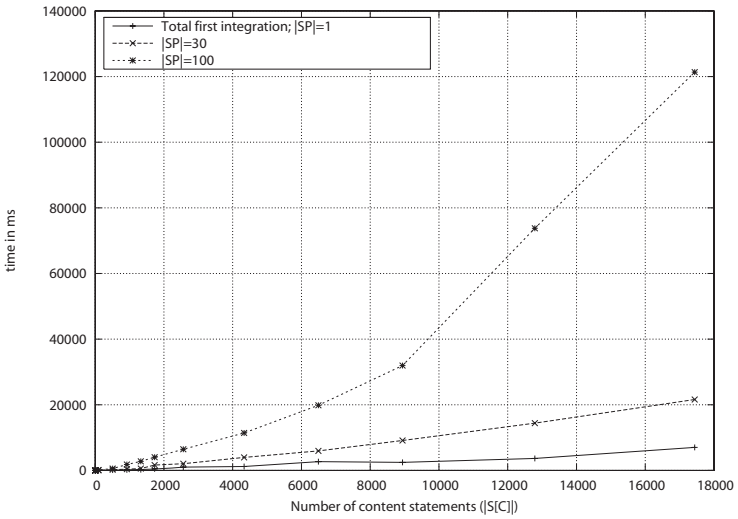


Figure A.2.: First Time Storage Annotation Times Increasing the Number of Peers

suggests that the retrieval of the swabbi-objects is little influenced by the number of content statements.

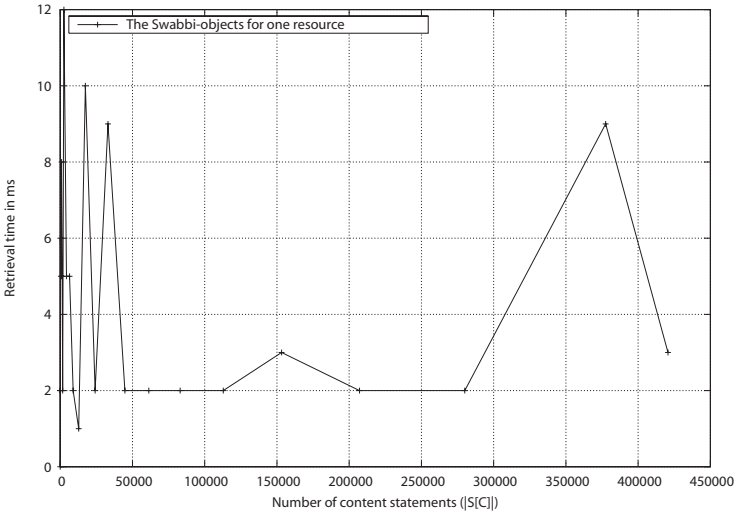


Figure A.3.: Time to Retrieve a Swabbi-object

Use Case 3 Figure A.4 plots the time to retrieve all swabbi peer-objects against the number of swabbi peer-objects, changing the number of content statements in the repository.

In summary the inclusion of the swabbi-objects in the repository is feasible. The time to include the statements into the repository doubles. With a growing number of peers refer-

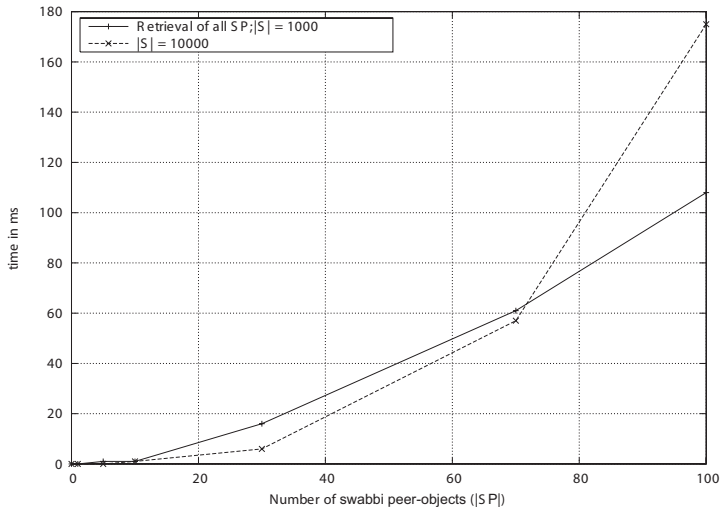


Figure A.4.: Time to Retrieve All Peers

enced in the LNR the time to retrieve these objects increases exponentially. The number of remote peers, which the routing algorithm is, however, low enough that this is uncritical.

Bibliography

Abecker, A. & van Elst, L. (2004). Ontologies for knowledge management. In Staab, S. & Studer, R. (Eds.), *Handbook on Ontologies*, pages 435–454. Springer.

Abecker, A., van Elst, L., & Dignum, V. (Eds.) (2004). *Proceedings of the 2nd Workshop on Agent-Mediated Knowledge Management at the 16th European Conference on Artificial Intelligence (ECAI'2004)*.

Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., & Schmidt, R. (2003). P-Grid: a self-organizing structured P2P system. *ACM SIGMOD Record*, 32(3):29–33.

Aberer, K., Cudre-Mauroux, P., Hauswirth, M., & van Pelt, T. (2004). GridVine: Building Internet-Scale Semantic Overlay Networks. In (van Harmelen et al., 2004), pages 107–121.

Adamic, L. A., Lukose, R. M., Puniyani, A. R., & Huberman, B. A. (2001). Search in power-law networks. *Physical Review E*, 64(046135):1–8.

Adar, E. & Huberman, B. (2000). Free riding on gnutella. *First Monday*, 5(10).

Aho, A. V., Denning, P. J., & Ullman, J. D. (1971). Principles of optimal page replacement. *Journal of the ACM*, 18(1):80–93.

Alavi, M. & Leidner, D. E. (2001). Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quarterly*, 25(1).

Albert, R. & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Albrecht, F. (1993). *Strategisches Wissensmanagement der Unternehmensressource Wissen*. Verlag Peter Lang.

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., & Werthimer, D. (2002). Seti@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61.

Androutsellis-Theotokis, S. & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371.

Arrow, K. J. (1963). *Social Choice and Individual Values*. Wiley, New York.

Arumugam, M., Sheth, A., & Arpinar, I. B. (2002). The peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web. In *Proceedings of the Workshop on Real World RDF and Semantic Web Applications at the 11th International World Wide Web Conference (WWW2002)*.

Aschoff, F., Schmalhofer, F., & van Elst, L. (2004). Knowledge mediation: A procedure for the cooperative construction of domain ontologies. In (Abecker et al., 2004), pages 20–28.

Babaoğlu, Ö., Meling, H., & Montresor, A. (2002). Anthill: A framework for the development of agent-based peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS2002)*, pages 15–22. IEEE Computer Society.

Balke, W.-T., Nejdl, W., Siberski, W., & Thaden, U. (2005). Progressive distributed Top-k retrieval in peer-to-peer networks. In *Proceedings of the 21st International Conference on Data Engineering (ICDE2005)*, pages 174–185. IEEE Computer Society.

Benjamin, P. C., Menzel, C., Mayer, R. J., Fillion, F., Futrell, M. T., DeWitte, P., & Lingineni, M. (1994). Ontology capture method (IDEF5). Technical report, Knowledge Based Systems, Inc.

Benjamins, V. R., Casanovas, P., Breuker, J., & Gangemi, A. (2005a). *Law and the Semantic Web*. Springer.

Benjamins, V. R., Casanovas, P., Contreras, J., Lopez-Cobo, J. M., & Lemus, L. (2005b). Iuriservice: An intelligent frequently asked questions system to assist newly appointed judges. In (Benjamins et al., 2005a), pages 201–217.

Berners-Lee, T. (1993). Naming and addressing: URIs, URLs, ... W3C Overview. available at http://www.w3.org/Addressing/.

Berners-Lee, T. (1998). Cool URIs don't change. W3C Style. available at http://www.w3.org/Provider/Style/URI.html.

Berners-Lee, T. (2000). Semantic Web. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html. Talk at the XML2000 conference.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Bol, G. (2004). *Deskriptive Statistik*. Oldenbourg.

Bonifacio, M., Bouquet, P., Danieli, A., Donà, A., Mameli, G., & Nori, M. (2004a). KEEx: A peer-to-peer solution for distributed knowledge management. In *Proceedings of the*

4th International Conference on Knowledge Management (I-KNOW'04), pages 43–52. Journal of Universal Computer Science (J.UCS).

Bonifacio, M., Camussone, P., & Zini, C. (2004b). Managing the KM Trade-Off: Knowledge centralization versus distribution. *Journal of Universal Computer Science*, 10(3):162–175.

Bonifacio, M., Bouquet, P., & Traverso, P. (2002). Enabling distributed knowledge management: Managerial and technological implications. *Novatica and Informatik/Informatique*, III(1):22–29.

Boose, J. H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition*, 1(1):3–37.

Breuker, J. & Winkels, R. (2003). Use and reuse of legal ontologies in knowledge engineering and information management. In *Proceedings of the Workshop on Legal Ontologies and Web-based Information Management at the 9th International Conference on Artifical Intelligence and Law (ICAIL2003)*.

Broekstra, J. (2003). SeRQL: Sesame RDF query language. Technical report, University of Karlsruhe. http://swap.semanticweb.org/public/Publications/swap-d3.2.pdf.

Broekstra, J., Ehrig, M., Haase, P., van Harmelen, F., Kampman, A., Sabou, M., Siebes, R., Staab, S., Stuckenschmidt, H., & Tempich, C. (2003). A metadata model for semantics-based peer-to-peer systems. In *Proceedings of the 1st Workshop Semantics in Peer-to-Peer and Grid Computing (SemPGRID) at the 12th International World Wide Web Conference (WWW 2003)*, pages 23–42.

Broekstra, J., Kampman, A., & van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF schema. In (Horrocks & Hendler, 2002), pages 54–64.

Buckingham Shum, S., Gangmin Li, V. U., Domingue, J., & Motta, E. (2003). Visualizing internetworked argumentation. In Kirschner, P. A., Shum, S. J. B., & Carr, C. S. (Eds.), *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making*, pages 185–204. Springer.

Buckingham Shum, S., Motta, E., & Domingue, J. (2002). Augmenting design deliberation with compendium: The case of collaborative ontology design. In *Proceedings of the Workshop on Facilitating Hypertext-Augmented Collaborative Modeling (HypACoM 2002) at the ACM Hypertext Conference*. Retrieved May 20, 2006 from http://kmi.open.ac.uk/projects/compendium/SBS-HT02-Compendium.html.

Buckingham Shum, S. & Hammond, N. (1994). Argumentation-based design rationale: what use at what cost? *International Journal of Human-Computer Studies*, 40(4):603–652.

Bussler, C., Davies, J., Fensel, D., & Studer, R. (Eds.) (2005). *Proceedings of the 2nd European Semantic Web Conference (ESWC2005)*. Springer.

Callan, J. P., Lu, Z., & Croft, W. B. (1995). Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28. ACM Press.

Camarinha-Matos, L. M. & Afsarmanesh, H. (Eds.) (2003). *Processes and Foundations for Virtual Organizations*. Kluwer Academic Publishers.

Casanovas, P., Casellas, N., Tempich, C., Vrandečić, D., & Benjamins, R. (2005). OPJK modeling methodology. In *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT) at the 11th International Conference on Artificail Intelligence and Law (ICAIL2005)*, pages 121–134. Wolf Legal Publishers.

Chen, P. P. (1976). The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

Cholvi, V., Felber, P., & Biersack, E. (2004). Efficient search in unstructured peer-to-peer networks. *European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services*, 15(6):535–548.

Christin, N., Weigend, A. S., & Chuang, J. (2005). Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *Proceedings of the 6th ACM conference on Electronic commerce (EC2005)*, pages 68–77. ACM Press.

Chu, W. W., Yang, H., Chiang, K., Minock, M., Chow, G., & Larson, C. (1996). Cobase: a scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 6(2-3):223–259.

Conklin, J. & Begeman, M. L. (1988). 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.

Conklin, J., Selvin, A., Shum, S. B., & Sierhuis, M. (2001). Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*, pages 123–124. ACM Press.

Cooper, B. F. (2004). Guiding queries to information sources with infobeacons. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware (Middleware '04:)*, pages 59–78. Springer.

Cooper, B. F. & Garcia-Molina, H. (2004). Sil: Modeling and measuring scalable peer-to-peer search networks. In *Proceedings of the 1st workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pages 2–16. Springer.

Cooper, B. F. & Garcia-Molina, H. (2005). Ad hoc, self-supervising peer-to-peer search networks. *ACM Transactions on Information Systems (TOIS)*, 23(2):169–200.

Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies: where is their meeting point? *Data & Knowledge Engineering*, 46(1):41–64.

Crespo, A. & Garcia-Molina, H. (2002a). Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 23–34. IEEE Computer Society.

Crespo, A. & Garcia-Molina, H. (2002b). Semantic Overlay Networks for P2P Systems. submitted for publication http://www-db.stanford.edu/~crespo/publications/op2p.pdf.

Cristani, M. & Cuel, R. (2005). A survey on ontology creation methodologies. *International Journal on Semantic Web Information Systems*, 1(2):49–69.

Cuenca-Acuna, F. M., Peery, C., Martin, R. P., & Nguyen, T. D. (2002). PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. Technical Report DCS-TR-487, Department of Computer Science, Rutgers University.

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

Daswani, N., Garcia-Molina, H., & Yang, B. (2003). Open problems in data-sharing peer-to-peer systems. In *Proceedings of the 9th International Conference on Database Theory (ICDT2003)*, pages 1–15. Springer.

Davenport, T. H. & Prusak, L. (1998). *Working Knowledge – How organisations manage what they know*. Havard Business School Press.

Davenport, T., De Long, D., & Beers, M. (1998). Successful knowledge management projects. *Sloan Management Review*, 39(2):43–57.

Davenport, T. H. (2005). *Thinking for a Living*. Havard Business School Press.

de Hoog, R. (1998). Methodologies for building knowledge based systems: Achievement and prospects. In *Handbook of Applied Expert Systems*, pages 1–14. CRS Press.

de Mántaras, R. L. & Saitta, L. (Eds.) (2004). *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*. IOS Press.

de Medeiros, A. P., Schwabe, D., & Feijó, B. (2005). Design rationale for model-based designs in software engineering. Monografias em Ciência da Computação 02/05, PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO.

De Nicola, A., Missikoff, M., & Navigli, R. (2005). A Proposal for a Unified Process for Ontology Building: UPON. In *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA 2005)*, pages 655–664. Springer.

Dean, M., Connolly, D., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2002). Owl web ontology language 1.0 reference. Technical report, W3C Working Draft.

Devedžić, V. (2002). Understanding ontological engineering. *Communications of the ACM*, 45(4):136–144.

Drucker, P. (1997). Looking ahead: Implications of the present. *Harvard Business Review*, 76(5):18–32.

Drucker, P. (1988). The coming of the new organization. *Harvard Business Review*, 66(1):45–53.

Easterbrook, S. (1991). Handling conflict between domain descriptions with computer-supported negotiation. *Knowlege Acquistion*, 3(3):255–289.

Ehrig, M. (2006). *Ontology Alignment: Bridging the Semantic Gap*. PhD thesis, Institut AIFB, Universität Karlsruhe (TH).

Ehrig, M., Schmitz, C., Staab, S., Tane, J., & Tempich, C. (2003). Towards evaluation of peer-to-peer-based distributed knowledge management systems. In (van Elst et al., 2003), pages 73–88.

Euzenat, J. (1995). 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. IOS Press.

Euzenat, J. (1997). A protocol for building consensual and consistent repositories. Rapport de recherche 3260, INRIA Rhône-Alpes, Grenoble (FR).

Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power-law relationships of the internet topology. In *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 251–262. ACM.

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

Fensel, D., Staab, S., Studer, R., van Harmelen, F., & Davies, J. (2003). A future perspective: Exploiting peer-to-peer and the semantic web for knowledge management. In *Towards the Semantic Web Ontology-based Knowledge Management*, pages 245–264. Wiley.

Fernández-López, M., Gómez-Pérez, A., Sierra, J. P., & Sierra, A. P. (1999). Building a chemical ontology using Methontology and the Ontology Design Environment. *IEEE Intelligent Systems*, 14(1):37–46.

Fox, G. (2001). Peer-to-peer networks. *Computing in Science & Engineering*, 3(3):75–78.

Gaines, B. R. (1989). Social and cognitive processes in knowledge acquisition. *Knowledge Acquisition*, 1(1):39–58.

Gangemi, A., Pisanelli, D., & Steve, G. (1998). Ontology integration: Experiences with medical terminologies. In (Guarino, 1998b), pages 163–178.

Gangemi, A., Sagri, M. T., & Tiscornia, D. (2003a). Metadata for content description in legal information. In *Proceedings of the Workshop on Legal Ontologies and Web-based Information Management at the 9th International Conference on Artifical Intelligence and Law (ICAIL2003)*.

Gangemi, A., Guarino, N., Masolo, C., & Oltramari, A. (2003b). Sweetening WORDNET with DOLCE. *AI Magazine*, 24(3):13–24.

Gil, Y., Motta, E., Benjamins, V. R., & Musen, M. A. (Eds.) (2005). *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*. Springer.

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of Intelligent Systems*, 16(3):391–409.

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

Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2003). *Ontological Engineering*. Advanced Information and Knowlege Processing. Springer.

Gong, L. (2001). Project JXTA: A technology overview. Technical report, Sun Microsystems Inc.

Gotel, O. & Finkelstein, A. (1994). An analysis of the requirements traceability problem. In *Proceedings of International Conference on Requirements Engineering 1994*, pages 94–101. IEEE CS Press.

Gravano, L. & García-Molina, H. (1995). Generalizing GlOSS to vector-space databases and broker hierarchies. In *Proceedings of the 21th International Conference on Very Large Databases, VLDB*, pages 78–89. Morgan Kaufmann.

Grüninger, M. & Fox, M. (1995). Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing at the 14th International Joint Conference on Artificial Intelligence (IJCAI1995)*.

Groove Networks (2001). Groove networks product backgrounder. White paper, Groove Networks. http://www.groove.net/pdf/backgrounder/GVO-Backgrounder.pdf.

Gruber, T. R. (1993a). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Gruber, T. R. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928.

Gruber, T. R. (1993b). Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers.

Grüninger, M. & Fox, M. (1995). TOVE: Manual of the Toronto Virtual Enterprise. Technical report, Department of Industrial Engineering, University of Toronto. available at http://www.ie.utoronto.ca/EIL/tove/ontoTOC.html.

Guarino, N. (1998a). Formal ontology and information systems. In (Guarino, 1998b).

Guarino, N. (Ed.) (1998b). *Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS)*. IOS-Press.

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

Haase, P., Broekstra, J., Ehrig, M., Menken, M., Mika, P., Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., & Tempich, C. (2004a). Bibster - a semantics-based bibliographic peer-to-peer system. In (van Harmelen et al., 2004), pages 122–136.

Haase, P., Siebes, R., & van Harmelen, F. (2004b). Peer selection in peer-to-peer networks with semantic topologies. In *Proceeding of the 1st International Conference on Semantics of a Networked World: Semantics for Grid Databases (ICSNW2004)*, pages 108–125. Springer.

Handschuh, S. (2005). *Creating Ontology-based Metadata by Annotation for the Semantic Web*. PhD thesis, Institut AIFB, Universität Karlsruhe (TH).

Hang Ng, C., Cheung Sia, K., & Chan, C.-H. (2003). Advanced peer clustering and firework query model in the peer-to-peer network. In *Proceedings of the 12th International World Wide Web Conference (WWW2003) - Posters*. ACM.

Hartmann, J., Sure, Y., Haase, P., del Carmen Suárez-Figueroa, M., Studer, R., Gómez-Pérez, A., & Palma, R. (2005). Ontology metadata vocabulary and applications. In *Proceedings of the Workshop on Web Semantics (SWWS) at the International Conference on Ontologies, Databases and Applications of Semantics (OTM2005)*, pages 906–915. Springer.

Herzog, O., Schek, H.-J., Fuhr, N., Chowdhury, A., & Teiken, W. (Eds.) (2005). *Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM2005)*. ACM Press.

Holsapple, C. W. (Ed.) (2003). *Handbook on Knowledge Management 2 – Knowledge Directions*. Springer, Heidelberg.

Holsapple, C. W. & Joshi, K. D. (2002). A collaborative approach to ontology design. *Communications of the ACM*, 45(2):42–47.

Hong, T. (2001). Performance. In (Oram, 2001), pages 203–241.

Horrocks, I. & Hendler, J. (Eds.) (2002). *Proceeding of the 1st International Semantic Web Conference (ISWC 2002)*. Springer.

House, E. (1980). *Evaluating with validity*. Sage Publications, Beverly Hills.

IEEE (1990). IEEE standard glossary of software engineering terminology. IEEE Standard 610.12-1990, ISBN 1-55937-067-X.

IEEE (1996). IEEE guide for developing of system requirements specifications. IEEE Standard 1233-1996.

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

Jones, D., Bench-Capon, T., & Visser, P. (1998). Methodologies for ontology development. In *Proceedings of the IT&KNOWS Conference of the 15th IFIP World Computer Congress*. Chapman-Hall.

Kalogeraki, V., Gunopulos, D., & Zeinalipour-Yazti, D. (2002). A Local Search Mechanism for Peer-to-Peer Networks. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 300–307. ACM Press.

Kalyanpur, A., Parsia, B., & Hendler, J. A. (2005). A tool for working with web ontologies. *International Journal on Semantic Web and Information Systems*, 1(1):36–49.

Kan, G. (2001). Gnutella. In (Oram, 2001), pages 94–122.

Karagiannis, D. & Reimer, U. (Eds.) (2004). *Proceeding of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM2004)*. Springer.

Keleher, P. J., Bhattacharjee, B., & Silaghi, B. D. (2002). Are virtualized overlay networks too much of a good thing? In *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems (IPTPS '01)*, pages 225–231. Springer.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843.

Kirkpatrick, S., Gelatt, C., & Vecci, M. (1983). Optimisation by simulated annealing. *Science*, 220(4598):671–680.

Kleinberg, J. (2000). Navigation in a small world. *Nature*, 406:845.

Kotis, K. & Vouros, G. A. (2005). Human-centered ontology engineering: The HCOME methodology. *Knowledge and Information Systems*.

Kuhn, T. S. (1996). *The Structure of Scientific Revolutions*. University of Chicago Press, 3rd edition.

Kunz, W. & Rittel, H. W. J. (1970). Issues as elements of information systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California.

Lamparter, S., Ehrig, M., & Tempich, C. (2004). Knowledge extraction from classification schemata. In *Proceeding of the International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE2004)*, pages 618–636. Springer.

Lassila, O. & Swick, R. (1999). Resource description framework (RDF). Proposed recommendation, W3C. http://www.w3c.org/TR/WD-rdf-syntax.

Lekatos, I. (1978). *The Methodology of Scientific Research Programmes*. Cambridge University Press.

Lenat, D. B. & Guha, R. V. (1990). *Building large knowledge-based systems. Representation and inference in the Cyc project.* Addison-Wesley, Massachusetts.

Leonard, D. (1995). *Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation*. Harvard Business School Press.

Li, Y., Bandar, Z. A., & McLean, D. (2003). An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882.

Liang, J., Kumar, R., Xi, Y., & Ross, K. (2005). Pollution in P2P file sharing systems. In *The 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*. IEEE.

Lladó, E., Salamanca, I., & Llodrà, B. (2002). D7.1 first user environment definition. SWAP project deliverable 7.1, Fundación IBIT.

Loo, B., Hellerstein, J., Huebsch, R., Shenker, S., & Stoica, I. (2004). Enhancing P2P file-sharing with an internet-scale query processor. In *Proceedings of the International Conference on Very Large Databases (VLDB2004)*, pages 432–443. Morgan Kaufmann.

Löser, A., Staab, S., & Tempich, C. (2005a). Semantic methods for P2P query routing. In *Proceedings of the 3rd German Conference on Multiagent System Technologies (MATES2005)*, pages 15–26. Springer.

Löser, A. & Tempich, C. (2005). On ranking peers in semantic overlay networks. In *WM 2005: Professional Knowledge Management - Experiences and Visions, Contributions to the 3rd Conference Professional Knowledge Management - Experiences and Visions*. DFKI, Kaiserslautern.

Löser, A., Tempich, C., Quilitz, B., Staab, S., Balke, W. T., & Nejdl, W. (2005b). Searching dynamic communities with personal indexes. In (Gil et al., 2005), pages 491 – 505.

Lozano-Tello, A. & Gómez-Pérez, A. (2004). ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15(2):1–18.

Lv, Q., Cao, P., Cohen, E., Li, K., & Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 2002 International Conference on Supercomputing (ICS 2002)*, pages 84–95. ACM Press.

Maedche, A., Motik, B., & Stojanovic, L. (2003). Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302.

Maedche, A. & Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79.

Maier, R. & Hädrich, T. (2004). Centralized versus peer-to-peer knowledge management systems. In *Proceedings of the 5th European Conference on Organizational Knowledge, Learning and Capabilities (OKLC)*.

Mann, W. C. (2005). RST Web Site. http://www.sfu.ca/rst.

Mann, W. C. & Thompson, S. A. (1988). Rhetorical Structure Theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Marcu, D. (1997). The rhetorical parsing of natural language texts. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, (ACL'97/EACL'97)*, pages 96–103. Morgan Kaufmann.

McGuinness, D. L. (2003). Ontologies come of age. In *Spinning the Semantic Web*, pages 171–194. MIT Press.

Merugu, S., Srinivasan, S., & Zegura, E. (2003). Adding Structure to Unstructured Peer-to-Peer Networks: The Role of Overlay Topology. In *Group Communications and Charges: Technology and Business Models. ICQT 2003 Proceedings*, pages 83 – 94. Springer.

Milgram, S. (1967). The small world problem. *Psychology Today*, 67(1).

Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., & Xu, Z. (2002). Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto.

Motro, A. (1990). FLEX: A tolerant and cooperative user interface to databases. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 2(2):231–246.

Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., & Risch, T. (2002). EDUTELLA: A P2P networking infrastructure based on RDF. In *Proceedings to the 11th International World Wide Web Conference (WWW2002)*, pages 604–615. ACM.

Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., & Löser, A. (2003). Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proceedings of the 12th International World Wide Web Conference, WWW2003*, pages 536–543. ACM.

Nonaka, I. & Takeuchi, H. (1995). *The Knowledge-Creating Company*. Oxford University Press, Oxford.

Nottelmann, H., Aberer, K., Callan, J., & Nejdl, W. (Eds.) (2005). *Proceedings of the 2005 ACM Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR 2005)*.

Noy, N. & Klein, M. (2003). Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440.

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

Noy, N. F. & Musen, M. A. (2002). The prompt suite: Interactive tools for ontology merging and mapping. Technical report, Stanford Medical Informatics, Stanford University, Stanford, California, USA.

Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71.

Oram, A. (Ed.) (2001). *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O´Reilly, Sebastopol (CA).

Paslaru Bontas, E. & Tempich, C. (2005). How Much Does It Cost? Applying ONTOCOM to DILIGENT. Technical Report TR-B-05-20, FU Berlin.

Pinto, H. S., Staab, S., Sure, Y., & Tempich, C. (2004a). OntoEdit empowering SWAP: a case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT). In *Proceedings of the 1st European Semantic Web Symposium ESWS2004*, pages 16–30. Springer.

Pinto, H. S., Staab, S., & Tempich, C. (2004b). DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In (de Mántaras & Saitta, 2004), pages 393–397.

Pinto, H. S. (2000). *Ontology Integration: Characterization of the Process and a Methodology to Perform it*. PhD thesis, UNIVERSIDADE TÉCNICA DE LISBOA INSTITUTO SUPERIOR TÉCNICO.

Pinto, H. S. & Martins, J. P. (2002). Evolving Ontologies in Distributed and Dynamic Settings. In *Proceedings of the 8th International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pages 365–374. Morgan Kaufmann.

Pinto, H. S. & Martins, J. (2001). A Methodology for Ontology Integration. In *Proceeding of the 1st International Conference on Knowledge Capture (K-CAP2001)*, pages 131–138. ACM Press.

Pinto, H. S., Tempich, C., Staab, S., & Sure, Y. (2005). Distributed Engineering of Ontologies (DILIGENT). In *Semantic Web and Peer-to-Peer*, pages 301–320. Springer.

Pinto, H. S. A. N. P. & Martins, J. P. (2004). Ontologies: How can they be built? *Knowlege Information Systems*, 6(4):441–464.

Plechawski, M. (2004a). D11.2 integrated platform with authorization. SWAP project deliverable 11.2, empolis Polska.

Plechawski, M. (2004b). D11.3 swap-orenge integration. SWAP project deliverable 11.3, empolis Polska.

Polanyi, M. (1966). *The Tacit Dimension*. Doubleday & Co., Garden City, NY.

Potts, C. & Bruns, G. (1988). Recording the reasons for design decisions. In *Proceedings of the 10th international conference on Software engineering*, pages 418–427. IEEE Computer Society Press.

Probst, G., Raub, S., & Romhardt, K. (1998). *Wissen managen*. Gabler Verlag, Wiesbaden.

Pujol, J. M., Flache, A., Sangüesa, R., & Delgado, J. (2004). Emergence of complex networks through local optimization. In (de Mántaras & Saitta, 2004), pages 48–52.

Ramakrishnan, N. & Grama, A. Y. (1999). Data mining-guest editors' introduction: From serendipity to science. *Computer*, 32(8):34–37.

Ramesh, B. & Dhar, V. (1992). Supporting systems development by capturing deliberations during requirements engineering. *IEEE Transactions on Software Engineering*, 18(6):498–510.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content addressable network. In *Proceedings of the Conference on applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM '01)*, pages 161–172. ACM Press.

Rowstron, A. & Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the International Conference on Distributed Systems Platforms (Middleware2001)*, pages 329–350. Springer.

Saroiu, S., Gummadi, P. K., & Gribble, S. D. (2003). A measurement study of peer-to-peer file sharing systems. *Multimedia Systems*, 9(2).

Schmitz, C., Staab, S., & Tempich, C. (2004). Socialisation in peer-to-peer knowledge management. In (Tochtermann & Maurer, 2004), pages 35–42.

Schmücker, J. & Müller, W. (2003). Praxiserfahrungen bei der Einfürung dezentraler Wissensmanagement Lösungen. *Wirtschaftsinformatik*, 45(3):307–311.

Schneider, U. (1996). Management in der wissensbasierten Unternehmung. In *Wissensmanagement*, pages 13–48. Frankfurter Allgemeneine Zeitung.

Schoder, D. & Fischbach, K. (2003). Peer-to-Peet Netzwerke für das Ressourcenmanagment. *Wirtschaftsinformatik*, 45(3):313–323.

Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing (P2P 2001)*, pages 101–102. IEEE Computer Society.

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

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Selvin, A., Shum, S. B., Sierhuis, M., Conklin, J., Zimmermann, B., Palus, C., Drath, W., Horth, D., Domingue, J., Motta, E., & Li, G. (2001). Compendium: Making meetings into knowledge events. In *Proceedings of the Knowledge Technologies Conference*.

Sharman Networks (2006). Kazaa. http://www.kazaa.com.

Shaw, M. & Gaines, B. (1989). Comparing conceptual structures: Consensus, conflict, correspondence and contrast. *Knowledge Acquisition*, 1(4):341–363.

Shirky, C. (2001). Listening to Napster. In (Oram, 2001), pages 21–37.

Siebes, R. (2002). Peer to peer solutions in the semantic web context: an overview. SWAP deliverable D1.1., Vrije Universiteit Amsterdam.

Skuce, D. (1995). Conventions for reaching agreement on shared ontologies. In *Proceedings of the 9th Banff Knowledge Acquisition Based Systems Workshop*, pages 3–19, Banff.

Skype Limited (2006). Skype. http://www.skype.com.

Sowa, J. F. (2000). *Knowledge Representation, Logical, Philosophical and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA.

Sripanidkulchai, K., Maggs, B., & Zhang, H. (2003). Efficient Content Location Using Interest Based Locality in Peer-to-Peer System. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*. IEEE.

Staab, S., Schnurr, H.-P., Studer, R., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34.

Staab, S., Studer, R., & Sure, Y. (2003). Knowledge processes and knowledge meta processes in ontology-based knowledge management. In (Holsapple, 2003), pages 47–68.

Stoica, I., Morris, R., Karger, D., Kaashoek, F., & Balakrishnan, H. (2001). Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (ACM SIGCOMM 2001)*, pages 149–160. ACM.

Stojanovic, L., Maedche, A., Motik, B., & Stojanovic, N. (2002). User-driven ontology evolution management. In *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW2002*, pages 285–300. Springer.

Störig, H. J. (1992). *Kleine Weltgeschichte der Philosophie*. Fischer Taschenbuch Verlag, Frankfurt am Main.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering principles and methods. *Data and Knowledge Engineering*, 25(1–2):161–197.

Sure, Y. (2003). *Methodology, Tools and Case Studies for Ontology based Knowledge Management*. PhD thesis, University of Karlsruhe.

Sure, Y., Angele, J., & Staab, S. (2003). Ontoedit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, 2800:128–152.

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

Sure, Y. & Studer, R. (2002). On-To-Knowledge methodology. In *On-To-Knowledge: Semantic Web enabled Knowledge Management*, pages 33–46. J. Wiley and Sons.

Sure, Y., Tempich, C., Vrandecic, D., & Pinto, S. H. (2004). Guidelines and evaluation for sekt ontology engineering. SEKT official deliverable 7.1.2, Institute AIFB, University of Karlsruhe (TH).

Susarla, A., Liu, D., & Whinston, A. B. (2003). Peer-to-peer enterprise knowledge management. In (Holsapple, 2003), pages 129–139.

Swartout, B., Patil, R., Knight, K., & Russ, T. (1996). Toward distributed use of large-scale ontologies. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*.

Tang, C., Xu, Z., & Dwarkadas, S. (2003a). Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (ACM SIGCOMM 2003)*, pages 175–186. ACM.

Tang, C., Xu, Z., & Mahalingam, M. (2003b). pSearch: information retrieval in structured overlays. *Computer Communication Review*, 33(1):89–94.

Tempich, C., Pinto, H. S., Staab, S., & Sure, Y. (2004a). A case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT). In (Tochtermann & Maurer, 2004), pages 225–232.

Tempich, C., Pinto, H. S., Sure, Y., & Staab, S. (2005a). An argumentation ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT). In (Bussler et al., 2005), pages 241–256.

Tempich, C. & Staab, S. (2005). Semantic query routing in unstructured networks using social metaphors. In Stuckenschmidt, S. & Staab, S. (Eds.), *Semantic Web and Peer-to-Peer*, pages 105–122. Springer.

Tempich, C., Staab, S., & Wranik, A. (2004b). REMINDIN': Semantic query routing in peer-to-peer networks based on social metaphors. In *Proceedings of the 13th International World Wide Web Conference, (WWW 2004)*, pages 640–649. ACM.

Tempich, C., Ehrig, M., Fluit, C., Haase, P., Martí, E. L., Plechawski, M., & Staab, S. (2004c). XAROP: A midterm report in introducing a decentralized semantics-based knowledge sharing application. In (Karagiannis & Reimer, 2004), pages 259–270.

Tempich, C., Ehrig, M., Staab, S., van Harmelen, F., Stuckenschmidt, H., Sabou, M., Siebes, R., & Broekstra, J. (2003). SWAP: Ontology-based knowledge management with peer-to-peer. In *Proceedings of the Workshop ontologiebasiertes Wissensmanagement (WM 2003)*, pages 17–20. Gesellschaft für Informatik.

Tempich, C., Löser, A., & Heizmann, J. (2005b). Community based ranking in peer-to-peer networks. In *Proceedings of the OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005*, pages 1261–1278. Springer.

Tempich, C., Pinto, H. S., & Staab, S. (2006). Ontology engineering revisited: an iterative case study with diligent. In *Proceedings of the 3rd European Semantic Web Conference (ESWC2006)*, pages 110–124. Springer.

Tempich, C., Pinto, H. S., Sure, Y., Vrandecic, D., Casellas, N., & Casanovas, P. (2005c). Evaluating DILIGENT ontology engineering in a legal case study. In *Proceedings of the 22nd World Congress - Law and Justice in a Global Society (IVR2005)*, pages 330–336. University of Granada.

Tempich, C. & Volz, R. (2003). Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In *Proceedings of the 2nd Workshop on Evaluation of Ontology-based Tools (EON2003) at the 2nd International Semantic Web Conference (ISWC 2003)*, volume 87. CEUR-Workshop proceedings http://ceur-ws.org.

The Economist (2006a). The new organisation: a survey of the company. *The Economist*, 378(8461). Survey.

The Economist (2006b). Taxonomy: Today we have naming of parts. *The Economist*, 378(8464). Survey.

Tochtermann, K. & Maurer, H. (Eds.) (2004). *Proceedings of the 4th International Conference on Knowledge Management (I-KNOW'04)*. Journal of Universal Computer Science (J.UCS).

Toulmin, S. (1958). *The Uses of Arguments*. Cambridge University Press.

Toulmin, S., Rieke, R., & Janik, A. (1984). *An introduction to reasoning*. Macmillan Publishing.

Tsoumakos, D. & Roussopoulos, N. (2003a). Adaptive probabilistic search for peer-to-peer networks. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P 2003)*, pages 102–109. IEEE Computer Society.

Tsoumakos, D. & Roussopoulos, N. (2003b). A comparison of peer-to-peer search methods. In *Proceedings of the International Workshop on Web and Databases (WebDB2003)*, pages 61–66.

Tsui, E. (2001). Technologies for personal and peer-to-peer (P2P) knowledge management. Technical report, CSC Leading Edge Forum (LEF).

Tsui, E. (2003). Tracking the role and evolution of commercial knowledge management software. In (Holsapple, 2003), pages 5–28.

Ullman, J. D. (2000). Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210.

Uschold, M. & Grüninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Sharing and Review*, 11(2):93–155.

Uschold, M., Healy, M., Williamson, K., Clark, P., & Woods, S. (1998a). Ontology Reuse and Application. In (Guarino, 1998b), pages 179–192.

Uschold, M. & King, M. (1995). Towards a methodology for building ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing at the International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998b). The enterprise ontology. *Knowledge Engineering Review*, 13(1):31–89.

Uschold, M. & Grüninger, M. (2004). Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64.

Uschold, M. (1996). Building ontologies: Towards a unified methodology. In *Proceedings of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*.

Valente, A. (2005). Types and roles of legal ontologies. In *Law and the Semantic Web*, pages 65–76. Springer.

van Elst, L., Dignum, V., & Abecker, A. (Eds.) (2003). *Agent-Mediated Knowledge Management International Symposium (AMKM 2003)*. Springer.

van Harmelen, F., McIlraith, S., & Plexousakis, D. (Eds.) (2004). *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*. Springer.

Visser, P., van Kralingen, R., & Bench-Capon, T. (1997). A method for development of legal knowledge systems. In *Proceedings of International Conference in Artificial Intelligence and Law (ICAIL1997)*, pages 151–160. ACM.

Voulgaris, S., Kermarrec, A.-M., Massoulie, L., & van Steen, M. (2004). Exploiting semantic proximity in peer-to-peer content searching. In *Proceedings of the 10th IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS'04)*, pages 238–243. IEEE.

Vrandecic, D., Pinto, H. S., Sure, Y., & Tempich, C. (2005). The diligent knowledge processes. *Journal of Knowledge Management*, 9(5):85–96.

Vrandecic, D., Sure, Y., Tempich, C., & Engler, M. (2006). Sekt methodology: Initial lessons learned and tool design. SEKT official deliverable 7.2.1, Institute AIFB, University of Karlsruhe (TH).

W3C (2001). World Wide Web Consortium (W3C) Semantic Web Activity Statement, available at http://www.w3.org/2001/sw/Activity/.

W3C (Ed.) (2004). *Proceedings of the 13th International World Wide Web Conference, (WWW 2004)*. ACM.

Watts, D. J. & Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393:440–442.

Wegner, H. (2002). Vorteile des einsatzes von agentenbasierten p2p-technologien im wissensmanagement. In *Proceedings of the 4. Konferenz zum Einsatz von Knowledge Management in Wirtschaft und Verwaltung (KnowTech 2002)*. ekkono GmbH. http://www.knowtech2002.de/wegner_ekkono_darmstadt.pdf.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49.

Wikipedia (2005). Wiki. http://en.wikipedia.org/wiki/WIKI.

Wikipedia (2006a). Routing. http://en.wikipedia.org/wiki/Routing.

Wikipedia (2006b). Voting systems. http://en.wikipedia.org/wiki/Voting_system.

Yin, R. K. & Campbell, D. T. (2003). *Case Study Research: Design and Methods*, volume 5. Sage Publications Inc., Thousand Oaks, CA.

Zaihrayeu, I. & Bonifacio, M. (Eds.) (2004). *Proceedings of the Workshop on Peer-to-Peer Knowledge Management (P2PKM 2004) at the MobiQuitous'04*, volume 108 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Zaihrayeu, I. & Robertson, D. (Eds.) (2005). *Proceedings of the Second Workshop on Peer-to-Peer Knowledge Management (P2PKM '05) at the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, volume 139 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Zimmermann, H. (1980). OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 28(4):425–432.

Please note: The cited URLs were provided by the author in all conscience. They were last checked in June 2005. However, even though "cool URIs don't change" (*cf.* (Berners-Lee, 1998)), URIs in the dynamic surrounding of the WWW (typically referred to as URLs (Berners-Lee, 1993)) and the content they represent are subject to change. In future they may differ from the cited sources in this work.

Index