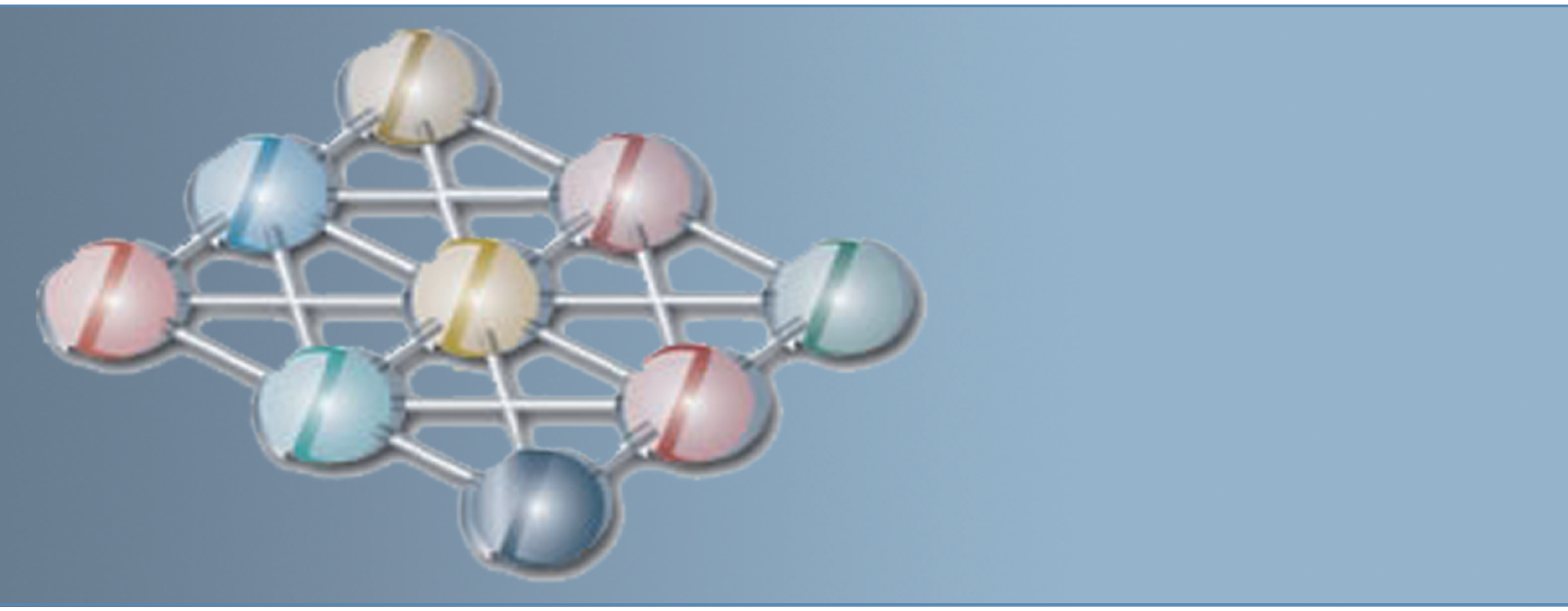
Peter Haase



Semantic Technologies for Distributed Information Systems

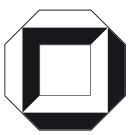
universitätsverlag karlsruhe

Peter Haase

Semantic Technologies for Distributed Information Systems

Semantic Technologies for Distributed Information Systems

by
Peter Haase



universitätsverlag karlsruhe

Dissertation, genehmigt von der Fakultät für Wirtschaftswissenschaften
der Universität Fridericiana zu Karlsruhe, 2006
Referenten: Prof. Dr. Rudi Studer
Prof. Dr. Christof Weinhardt
Prof. Dr. Frank van Harmelen

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2007
Print on Demand

ISBN: 978-3-86644-100-2

Contents

List of Figures

List of Tables

Part I

Foundations

Chapter 1

Introduction

1.1 Motivation

The effective management of information has become a key factor of many aspects of our every day life. Over the past years, there has been tremendous growth in quantity of data being produced. Recent studies indicate that the data production is growing at around 50 percent compound annual growth rate [VL] with several exabytes (10^{18}) of data being generated each year. The management of large amounts of data, per se, is not a very difficult problem anymore. Data warehouses tend to easily exceed one terabyte (10^{12}) in size. While we have the technology to store and access data, we seem to lack the ability to transform data into useful information and extract knowledge from them. In fact, today data consumption is slower than production. The problem thus lies in the question of how to make use of the data: As long as no proper interpretation for the data exists, the data is of no use – a situation commonly called information starvation. Semantics is arguably the single most important ingredient in enabling information systems to overcome this situation.

Another important aspect of today's information management is the increasing degree of distribution: Data is no longer stored on one logical server (such as in a well-architected warehouse), but is distributed across multiple nodes in different organizations, some within and some across enterprises. In recent years, we have seen a remarkable development of distributed information systems, both in terms of quantity, i.e. the scale of participating nodes and the amount of data that is being managed, as well as in terms of the types of systems being realized.

Distributed information systems exhibit characteristics that pose particular challenges for their realization:

1. heterogeneity, i.e. discrepancies in the way the data is modeled and represented,
2. dynamics, i.e. changes in the structure of the system as well as changes in the information at the individual nodes, and

3. autonomy, i.e. the ability of nodes to take decisions independently.

The integration of heterogeneous data sources is a problem that has been investigated for many years. However, the thorny question of semantic heterogeneity remains. Any two schemas designed by different individuals will never be identical. They will have different units (one price is in euros, another is in dollars), different semantic interpretations (one price is net, another is gross), and different names for the same thing (Alon Y. Levy in one database, but Alon Halevy in another). A semantic heterogeneity solution capable of deployment at large scale remains elusive.

Further, distributed information systems are subject to continuous change and evolution. Rarely will we find application scenarios that are fixed and static over their lifetime. Reasons for such changes are manifold: In a world that is constantly changing, we need to deal with changes in application environments, domain changes, changes in the conceptualization of the domain, or the adaptation to different tasks. In order for the data to remain useful, i.e. to consistently interpret the data in such dynamic settings, effective means for change management and evolution support are required.

At the same time we are experiencing a drift from centralized to decentralized architectures. This implies that in modern types of distributed information systems, we need to deal with an increasing degree of autonomy of participating nodes. Traditional approaches found, for example, in federated databases assumed a central instance of control. While this may be a feasible approach within an enterprise with a few dozen operational systems, traditional paradigms completely break if we need to perform integration among different enterprises, or on an ad hoc basis. Peer-to-Peer systems that form large networks of nodes without centralized control and more recently Grid systems that allow to form virtual organizations promise new paradigms, but also provoke new technical challenges.

Semantic technologies are expected to provide a suitable framework to deal with the heterogeneity, dynamic nature and massive scale of autonomous resources in modern distributed information systems. With semantic technologies we refer the application of techniques that support and exploit semantics of information (as opposed to syntax and structure) to enhance existing information systems. Ontologies play a particularly important role in the use of semantic technologies. An ontology characterizes a domain of discourse by identifying concepts and relationships between them in a formal language. As such, ontologies allow to interpret data consistently across heterogeneous nodes and thus support information integration. Further, the techniques of ontology evolution provide the means to deal with the dynamics of information in a consistent manner. Finally, semantic descriptions based on ontologies allow to coordinate autonomous resources in large scale distributed information systems.

1.2 Research Questions and Contributions

The main goal of this work is to answer the question:

Main Question *How can the use of semantic technologies based on ontologies address the challenges arising from the specific characteristics of distributed information systems?*

Answering this question requires a careful analysis of what exactly distributed information systems are. By looking at specific classes of systems, including traditional ones such as federated databases, but also more recent ones such as Peer-to-Peer and Grid systems, we are able to identify common characteristics and resulting challenges.

Examining these challenges, we can refine the main question into more specific questions. As answers to these questions, we provide solutions that are integrated in a framework of semantic technologies for distributed information systems. A main goal for this framework is the adherence to emerging standards in the field of semantic technologies. In particular, we base our entire work on the formal ontology model of OWL as the standard language for representing ontologies. The contributions of this work are thus made directly applicable to a wide variety of application scenarios.

The specific research questions and contributions can be grouped by the characteristics of heterogeneity, dynamics, and autonomy. To address the characteristics of heterogeneity, we ask the following questions:

Question I.1 *How can we express the rich semantic relationships between heterogeneous data sources described using ontologies in the absence of a single, global ontology?*

The problem of data integration is arguably one of the most extensively studied problems in distributed information systems. The solution that we present falls into the line of existing work in this area. The main contribution of our mapping system is its expressiveness in two important dimensions: first, the expressiveness of the ontologies that are to be mapped, i.e. we support the rich ontology language OWL, and second the expressiveness of the mappings themselves.

Question I.2 *How can we formally grasp the problem of measuring similarity between heterogeneous ontologies?*

As answer to this question we present a similarity framework for ontologies that we describe in terms of the OWL ontology language. The contribution of this framework consists in its comprehensiveness and applicability to a broad range of purposes, in particular integration tasks.

Further, we address two important questions regarding the dynamic nature of distributed information systems:

Question II.1 *How can we guarantee a useful interpretation of information by ensuring the consistent evolution of ontologies in a dynamic environment?*

As ontologies in real-world distributed information systems are typically not static entities, but change over time, one of the major problems is the potential introduction of inconsistencies as a result of applying changes. While the problem of dealing with evolution has been addressed in various related areas, including database schema evolution, the problem of evolving ontologies is largely unexplored. Our approach comprises novel methods for the consistent evolution of OWL ontologies according to various consistency conditions. Further, we compare the approach of consistent ontology evolution within a framework for handling inconsistencies in changing ontologies.

Question II.2 *Can we exploit the heterogeneity of individual nodes in a collaborative scenario to pro-actively recommend adaptations to ontologies?*

Dealing with ontology changes in a network of heterogeneous ontologies implies additional challenges. But in turn, we can see the very existence of the frequent changes in such networks of ontologies not only as complication, but as a source for discovering changes that might be useful for other nodes in the network. We demonstrate this with a novel technique of adapting collaborative filtering for change recommendations in ontology evolution.

Finally, the following questions relate to the autonomy of resources in distributed information systems:

Question III.1 *How can we semantically describe autonomous resources in distributed information systems in order to support coordination tasks?*

Metadata ontologies to semantically describe resources have already been proven to be useful in many applications. The contribution of our metadata ontology consists in the coverage of characteristics of autonomous resources in open distributed information systems. We complement this work with a Peer-to-Peer application for the decentralized management of metadata descriptions.

Question III.2 *How can we use semantic descriptions of resources in order to solve the coordination task of resource selection in a network of autonomous nodes?*

The problem of resource discovery and selection is critical especially in the absence of centralized control. As a solution to the problem, we present a novel approach of building a semantic overlay network based on the semantic descriptions of resources, which serves as the basis for an intelligent routing of requests. Extensive evaluations demonstrate the benefit compared to alternative approaches.

The final contribution consists in the fact that most of the methods and algorithms presented in the work have been implemented in different applications and evaluated both in in-vitro and real-life experiments to show the benefit of the application of semantic technologies. Most notably, a large part of the work has been realized in the award-winning Bibster system.

In the next section, we give an outline of the book that clarifies the relation between the chosen structure and the motivation and contributions mentioned above.

1.3 Reader's Guide

The book is organized into four main parts as shown in Figure 1.1. The arrows indicate dependencies between the individuals parts and chapters. While Part I provides an overview of important foundations for this work, the subsequent three parts each cover solutions for the three main challenges identified in distributed information systems.

Part I - Foundations In Chapter 2 we provide an overview of distributed information systems. We introduce the reader to ontologies – in particular to the ontology language OWL and related languages – in Chapter 3. In Chapter 4 we provide an overview of how semantic technologies based on ontologies can be employed in distributed information systems. In Chapter 5 we present one concrete system – Bibster – as application scenario that serves as running example throughout this work to illustrate the importance and applicability of the presented work.

Part II - Ontology-based Information Integration In this part we develop techniques to deal with the problem of heterogeneity in distributed information systems. In Chapter 6 we develop a mapping system for distributed ontologies. This mapping system deals with the representation of complex correspondences between OWL-based ontologies and provides reasoning support for query answering against integrated ontologies. In Chapter 7 we address a related problem, which can be seen as complementary to that of mapping representation: We present a framework for measuring similarity in and between heterogeneous ontologies.

Part III - Ontology Evolution In this part we turn to the second important aspect of distributed information systems – their dynamic nature – and provide corresponding solutions. In Chapter 8 we address the important issue of consistent evolution of OWL ontologies. We consider the scenario of collaborative evolution in Chapter 9.

Part IV - Ontology-based Coordination In this final part we present specific solutions for coordinating distributed systems consisting of autonomous nodes. In Chapter 10 we propose a metadata ontology for the purpose of describing resources in distributed information systems. In Chapter 11 we turn to a specific coordination task, where we rely on the semantic descriptions to form a semantic overlay network for peer selection and query routing.

We conclude the book with a summary, an overview of open questions and an analysis of interesting future directions in Chapter 12.

1.4 Publications

Large parts of this work have been published before in a number of workshop and conference papers, journal articles, and book chapters. The work is the result of fruitful cooperations in the EU projects SWAP[1] and SEKT[2]. In the following list we provide references to relevant publications for the individual chapters of this book.

- Most of Chapter 5 has been published before in various publications about the Bibster system [HBE+04, HSB+04] and the SWAP architecture [EHvH+03]. The Bibster system was developed in collaboration with the project partners of the SWAP project.
- The mapping system for the integration of OWL ontologies presented in Chapter 6 was first published in [HM05].
- A previous version of the similarity framework of Chapter 7 was presented in [EHSH05].
- The methods for consistent evolution of OWL ontologies in Chapter 8 have first been published in [HS05a]; the framework for comparing the alternative approaches for dealing with inconsistencies was originally presented in [HvHH+05].
- Chapter 9 is largely based on work on collaborative ontology evolution first presented in [HHSTS05].
- The work on ontology-based coordination in Chapter 10 is partially based on publications about the ontology metadata vocabulary OMV [HPS+05], [HSH+05] and the Oyster system [PH05].
- Chapter 11 is based on two prior publications. The model of expertise based peer selection along with simulation experiments was first published in [HSvH04]. The results of the experiments with the Bibster system were presented in [HSB+04].

[1] http://swap.semanticweb.org/
[2] http://www.sekt-project.com/

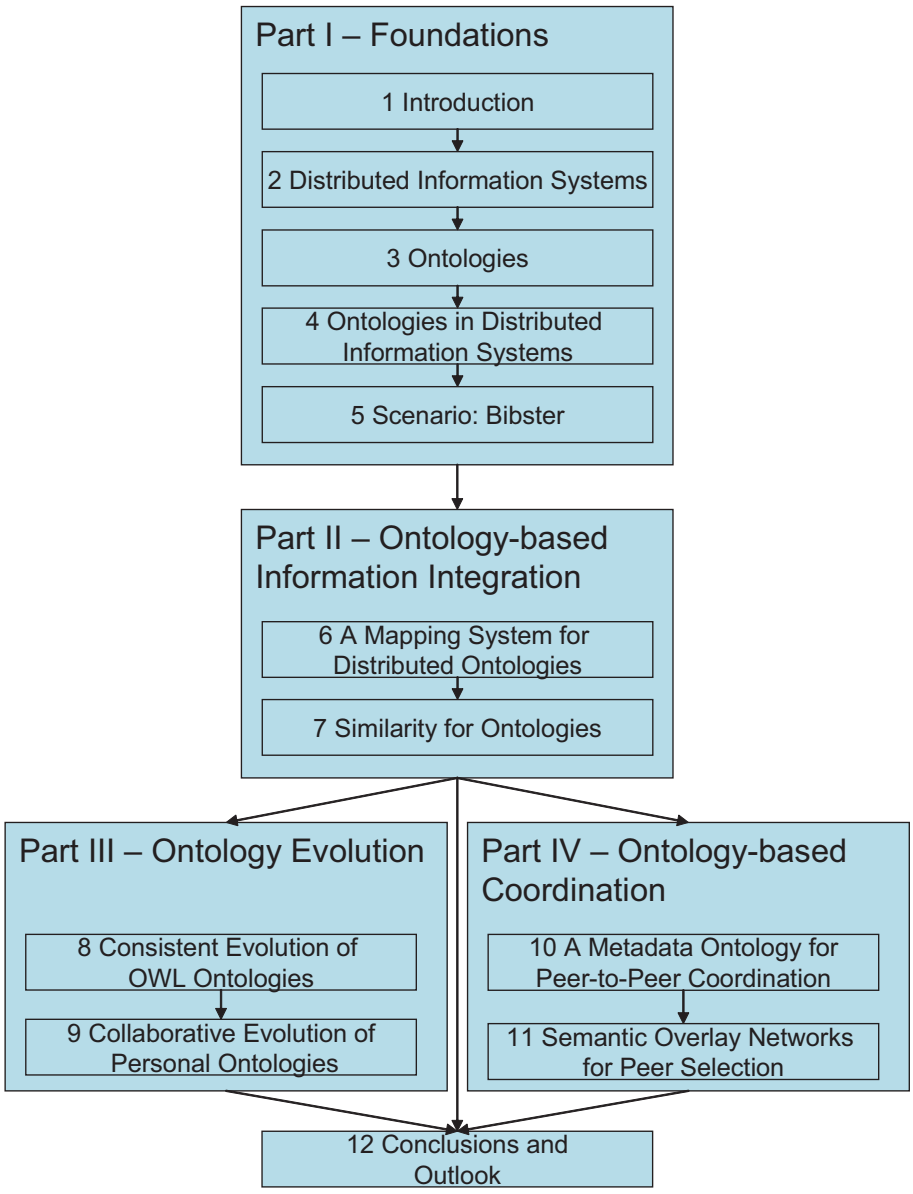


Figure 1.1: Structure of the Book

Chapter 2

Distributed Information Systems

In this chapter we discuss the concept of a *Distributed Information System*. In Section 2.1 we define important terms for the discussion. We then analyze particular classes of established and emerging Distributed Information Systems: Distributed and Federated Databases in Section 2.2, Peer-to-Peer systems in Section 2.3, and Grid systems in Section 2.4. In Section 2.5 we identify three common characteristics along this categorization that pose particular challenges when building distributed information systems: heterogeneity, dynamics and autonomy.

2.1 Terminology

A serious obstacle to any in-depth discussion of distributed information systems is the lack of an agreed terminology. As a wide spectrum of information systems is classified as distributed, a universally accepted definition is hard to provide. We therefore discuss and define the key concepts and terms associated with distributed information systems as they are used throughout this work.

Before we talk about *Distributed* Information Systems, we will answer the question what an *Information System* is. Inspired by Aberer [Abe06], we can define it as follows: *"An information system is a system that manages facts about some domain for a specific purpose."* This definition contains a number of concepts that require explanation:

- The *facts* refer to the actual data that is managed by the system. The data is represented using some mathematical structure, called the *data structure*, which describes how the data is organized.
- The *domain* refers to some universe of discourse, which is not necessarily our physical real world.
- *Managing* means to maintain, i.e. to persistently store the data, as well as to perform operations on the data, such as processing, presenting and disseminating

the data. The amount of data that is stored is usually very large, which poses serious problems for efficiency. This aspect of information systems is covered by data management systems.

- The *purpose* refers to the fact that every information system has an entity – which may be human or a computer – that makes use of it. The intention of that entity is to perform a certain task related to some aspect of the real world, e.g. making a decision, performing a computation etc. In order to do so, the entity performs an *interpretation* of the data. The interpretation relates the data structure (also called the syntax) with the semantics, i.e. it assigns a meaning to the data. This interpretation can be a formalized relationship, or just exist informally.

There exists a lot of confusion between the notions of *data* and *information* management. The reason is that the two issues are inherently connected and that the distinction is mainly a matter of emphasis: Data management puts emphasis on efficiently and reliably managing large amounts of data and syntactic aspects. However, any database management system can be used as an information system. Information management puts emphasis on managing complex interpretations of data, semantic aspects. In turn, any information system requires data management.

Distributed Information Systems refer to information systems where the data is physically distributed across multiple nodes, but is somehow logically related. There may be many different reasons why data may be distributed. Some of these reasons are related to the improved use of existing distributed physical resources. We might want to move data close to the node where it is processed (locality of access), we might take advantage of parallel processing of the data, and we might want to avoid bottlenecks in order to improve scalability of the system. Thus, we *want to distribute* the data. In other cases it may be that the data *is already distributed* to begin with, i.e. we want to enable the interoperability of (pre-existing) information systems, or to allow different interpretations of the same data for different needs and capabilities.

The term *distributed* is often mistaken to imply *decentralized*. However, this is not the case. In fact, most distributed systems today are centralized systems: A typical example are client/server architectures, which are distributed, but rely on a server as a centralized node.

The difference between centralized and decentralized systems is essentially that of their topology, i.e. how the nodes in the system are connected [Min01]. Figure 2.1 depicts a simple graphical representation of centralized vs. decentralized topologies. While in centralized topologies there is one (or multiple) distinguished node to coordinate the system, in decentralized topologies such a distinguished node does not exist. The Internet itself is in its roots completely decentralized. However, most applications built on this infrastructure follow the centralized client/server paradigm. The distinction between centralized and decentralized topologies is not a crisp one. There are many degrees in between, e.g. various forms of *hybrid topologies* [DNB05].

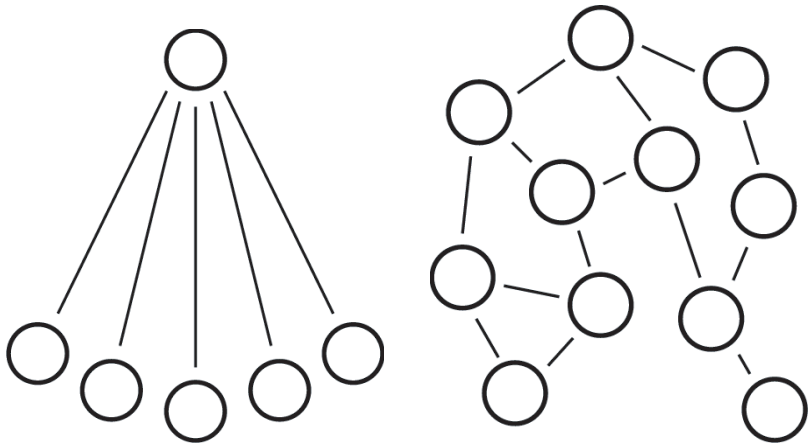


Figure 2.1: Centralized and Decentralized Topologies

In the following we present three typical classes of distributed information systems. By no means these classes are to be seen as disjoint classes of distributed information systems. They rather represent different paradigms being researched and applied in partially disjoint communities and areas, but share a large degree of overlap.

2.2 Federated Databases

The history of database systems is a prime example for the historic developments of distributed information systems. Looking at this history in some detail helps to understand the characteristics of distributed information systems in general.

The roots of modern database systems date back to the early 1950s and 60s, where the first ideas of storing data records in files as part of a simple file system and linking data records to represent more complex data structures evolved [Sim96]. In the late 60s Tedd Codd developed the *relational model* [Cod70], which became the basis of much of relational database theory and technology, and subsequently dominated over competing models such as the hierarchical and network data model. While first database systems assumed that all data is managed in a single database, the need for addressing the challenges of managing distributed data became apparent rather early. Today, we can look back to almost 30 years of distributed data management, but by no means all problems related to distributed data management have been solved.

Generally, in database systems we can distinguish between several forms of distribution. In the most simple form of *distributed databases*, the data is distributed – either replicated or partitioned – across multiple nodes, but the schema is globally shared. The

use of a global schema becomes possible, because in distributed database systems the data typically is *deliberately* distributed, either for increased performance or reliability.

On the other end of the spectrum, *federated databases* allow heterogeneity of the schema across the nodes and an increased degree of local autonomy. The term *federated database system* has first been coined by Hammer and McLeod [HM78]. Since then the term has been used for several different but related architectures of database management systems. We here use term in its broader context and most prominent meaning as presented by Sheth and Larson in [SL90]: A *federated database system* is a collection of cooperating but autonomous component database systems. Federated database systems can be characterized and classified according to their heterogeneity and autonomy:

Heterogeneity in federated databases mainly arises from differences in the individual database management systems and in the semantics of the data. Differences in the database management systems are called system heterogeneity and may include heterogeneity in the supported data models, data structures, and query languages, as well as in system aspects, including differences in the transaction management and communication capabilities. Semantic heterogeneity arises if there is disagreement about the meaning and interpretation of the data. In the context of federated databases, semantic heterogeneity is often referred to as schema heterogeneity, i.e. differences in how the data is modeled. However, addressing schema heterogeneity alone does not sufficiently solve the problem of semantic heterogeneity, as the schemas themselves often do not provide enough semantics to interpret the data consistently.

Autonomy here refers to the fact that a node can continue its local operation and at the same time participate in the federation. Often the entities managing the different database management systems are under separate control. Design autonomy is the most important form of autonomy in federated database systems, as it requires to deal with the different capabilities of the participating data sources [LYV+98]. The autonomy of database systems to freely join and leave the federation (association autonomy) is typically restricted in the interest of manageability, as this would require the federated database system to be designed such that its existence and operation are not dependent on a single component. Dealing with the autonomy in federations becomes a significant challenge with a large number of data sources.

The component database systems may be integrated to various degrees. Depending on whether a global schema is used for integrating the local schemas, we can further distinguish between *tightly coupled* or *loosely coupled* federated database systems, respectively. A reference architecture for tightly coupled federated databases based on a global federated schema is described in [SL90]. Mediator architectures [Wie92] – as an example for loosely coupled architectures – do not rely on a global schema. Instead mediators form a distinct middle layer between user applications and data resources. Mediators exploit specific knowledge about certain sets of data to be able to select appropriate data resources, as well as to abstract and transform queries and resulting data.

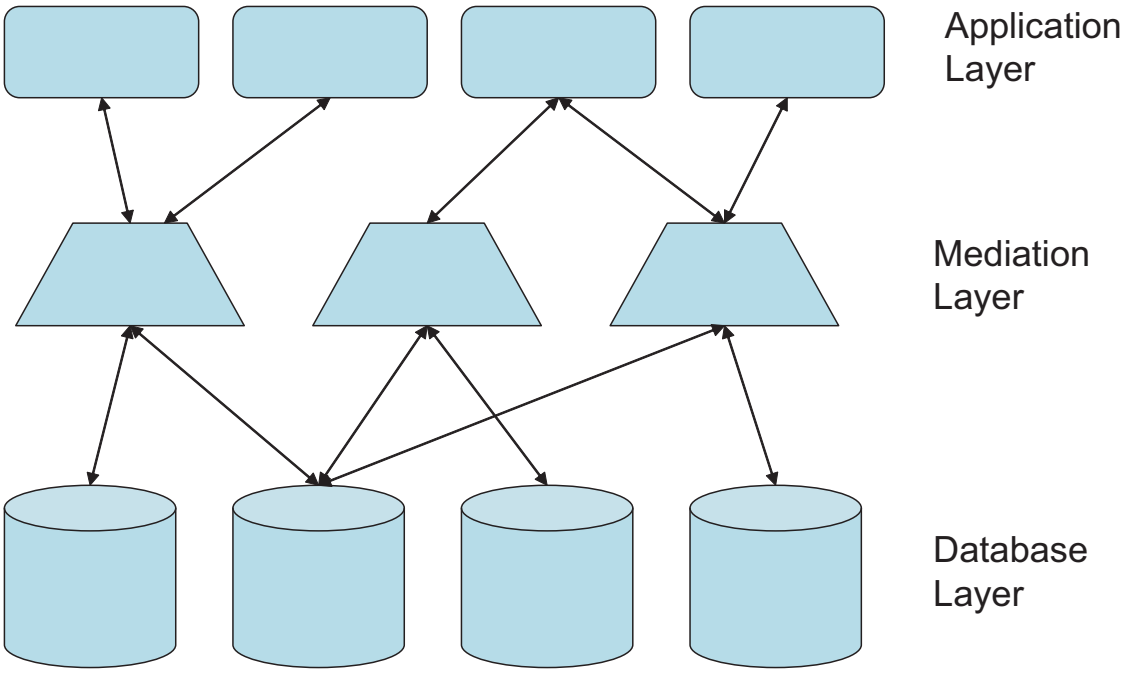


Figure 2.2: Mediator Architecture

A typical mediator architecture is presented in Figure 2.2. It shows the role of the mediators as an intermediary layer between applications and data sources.

An additional problem that frequently occurs in federated database systems is that of handling schema changes on the individual nodes of the federation. Schema evolution and updates to databases are already challenging problems for single database systems. In a federated database management the complexity increases as changes need to be reflected either in the mapping with the global federated schema (in the case of a tightly coupled system) or in the logics of the mediator (in the case of a loosely coupled, mediator based system).

Example Systems. Early examples of realizations of federated databases are surveyed in [SL90], including MRDSM, OMNIBASE and CALIDA as examples for loosely coupled systems and SIRIUS-DELTA and DDTS as examples for tightly integrated systems. More recent systems include Garlic [RS97] and TSIMMIS [LYV+98]. Current developments include the InfoSphere project at IBM, whose goal is the automation of federations of tens of thousand data sources [BHM+05].

Recently, Peer-to-Peer database systems have received increasing attention. In a sense, they can be seen as the logical next step following loosely coupled federated databases in terms of the degree of decentralization and local autonomy.

2.3 Peer-to-Peer Systems

Peer-to-Peer systems have emerged as a new paradigm for distributed information systems, in which peers (or nodes) share data – or other resources – via the internet or private networks. In these systems the nodes share resources in a way that every peer can act as a client and server, i.e. requester and provider of resources. We can further analyze the characteristics of Peer-to-Peer systems along three underlying principles, as they have been introduced in [AH02]:

- *The principle of sharing resources:* All Peer-to-Peer systems involve an aspect of resource sharing, where resources can be physical resources, such as disk space or network bandwidth, as well as logical resources, such as services or different forms of knowledge. By sharing resources applications can be realized which could not be set up by a single node. The peers themselves decide when and which resources are to be shared.

- *The principle of decentralization:* There typically exists no central node for coordination. Instead, there is a direct communication between the individual nodes. This principle is an immediate consequence of sharing resources. Decentralization is in particular interesting in order to avoid a single point of failure or performance bottlenecks in the system.

- *The principle of self-organization:* Due to the lack of a central node to coordinate the activities of the system, nodes have to self-organize, based on whatever local information is available and interacting with locally reachable nodes (neighbors). The global behavior then emerges as the result of all the local behaviors that occur.

The terminology around Peer-to-Peer systems is sometimes confusing and often inconsistent, as many famous "Peer-to-Peer"-systems such as Napster often have centralized or hybrid topologies. To further complicate things, the term Peer-to-Peer often refers to different layers of interaction. [HS05b] introduces a three-layer model to describe the various types of possible interactions in Peer-to-Peer systems. According to this model, the Peer-to-Peer paradigm can be applied to infrastructure, applications, and communities:

1. *Peer-to-Peer Infrastructure.* This layer refers to systems that manage the communication, integration, transformation, identification and discovery of resources in a Peer-to-Peer fashion. The shared resources may for example be information items, leading to Peer-to-Peer information management systems.

2. *Peer-to-Peer Applications.* Even if a system relies on a centralized infrastructure, it may be possible that the Peer-to-Peer paradigm is applied at the application

level. For example, message brokering systems often allow a direct communication between nodes on the application level (thus departing from the client/server paradigm), while on the infrastructure level the communication is still realized using centralized message brokers.

3. *Peer-to-Peer Communities.* Finally, systems may allow the users to interact in Peer-to-Peer fashion. The ideas of Peer-to-Peer communities are increasingly found in what is informally called Web 2.0 [O'R05], embracing the power of the web to harness collective intelligence of online communities: Wikipedia[1], an online encyclopedia based on the notion that an entry can be added by any web user, and edited by any other, is one such approach to content creation. Sites like del.icio.us[2] and Flickr[3] have pioneered a concept called "folksonomy" (in contrast to taxonomy), a style of collaborative categorization of sites using freely chosen keywords, often referred to as tags.

The application of the Peer-to-Peer paradigm to distributed information systems mainly concerns the infrastructure layer. The advantages of such Peer-to-Peer infrastructures over centralized approaches are manifold:

- The distribution allows scalability both in data volumes and the number of connected parties.
- The costs for starting and maintaining such systems are generally low, which implies a low entrance barrier.
- The decentralized and often self-organized coordination holds the potential of failure tolerance, i.e. robustness against failure of any single component and intentional attacks, avoiding a bottleneck for both computational performance and information update.
- New resources can be dynamically integrated.
- Finally, new application scenarios are realizable which were not possible with centralized architectures.

However, these advantages do not come for free: The large degree of distribution of Peer-to-Peer systems is also the cause of a number of new problems: As there typically exists no central node for coordination and instead there is a direct communication between the individual nodes, one of the biggest challenges is enabling nodes to find one another, to construct a network topology, and to efficiently route requests in a scalable

[1] `http://www.wikipedia.org/`
[2] `http://del.icio.us/`
[3] `http://www.flickr.com/`

manner. Further, peers can freely leave and join the network, which poses challenges for the availability and reliability of the system.

The first successful Peer-to-Peer information systems were systems whose primary focus was that of very simple resource sharing. These resources typically were files, and search for resources was restricted to simple keywords or file names. As Peer-to-Peer systems evolve from simple file sharing systems to support structured and semantically rich data, several additional information management issues need to be addressed. Specifically, Peer-to-Peer systems need to deal with the location of information, information integration, query processing, and consistency issues [SAB+05]. In particular, the lack of a single coherent schema for organizing information sources across the Peer-to-Peer network hampers the formulation of search queries. Duplication of information across the network results in many duplicate answers to a single query, and answers to a single query often require the integration of information residing at different, independent and uncoordinated peers.

Additionally, we are faced with the issue of dealing with the dynamics of changing schemas and information at the individual peers. The problem here is considerably harder than for example in federated database systems because of the higher degree of decentralization: While in federated databases the problem can be addressed on the level of global schema or the mediator, these concepts are typically not present in Peer-to-Peer systems, such that changes need to be propagated in a decentralized way.

Example Systems. The term Peer-to-Peer has become famous with large-scale file sharing systems such as Napster, Gnutella [Kan99], and Freenet [CMH+02]. A more detailed overview of these specific systems can also be found in [AH02]. These systems are fairly simple in terms of the descriptions of their resources. They typically limit the search to filenames. Instead, most effort has been taken to achieve scalable solutions. The issue of decentralized coordination schemes for robust and efficient routing of queries is the topic of a number of Peer-to-Peer systems in the research community, including CAN [RFH+01], Chord [SMK+01], Pastry [RD01], and P-Grid [ACMD+03]. A more detailed discussion of these systems follows in Chapter 11.

Besides the question of the organization of the network and efficient query routing, there are several systems that concentrate on other aspects of Peer-to-Peer information management. For example, Piazza [TIM+03] focuses on the issue of representing local mappings between heterogeneous peers. DBGlobe [PAP+03] aims at developing a Peer-to-Peer data management system for modeling, indexing and querying data hosted by massively distributed, autonomous and possibly mobile peers. DBGlobe is centered around managing XML data; it provides (1) infrastructure support, including mobile peers and the creation of context-dependent communities, (2) metadata management for services and peers, (3) filters for efficiently routing path queries on hierarchical data,

and (4) querying using the AXML language, an XML query language that incorporates service calls.

Finally, there are general purpose Peer-to-Peer infrastructure projects. JXTA [TAD+02] – being the most prominent one – attempts to provide a uniform interface to Peer-to-Peer systems and to facilitate interoperability. JXTA defines a three layer Peer-to-Peer software architecture (applications, services, core), a set of XML-based protocols, and a number of abstractions and concepts such as peer groups, pipes and advertisements.

2.4 Grid Systems

Grid systems have their roots in distributed computing, rather than distributed information systems. In distributed computing, the focus traditionally is on the computational complexity rather than data complexity. However, Grid computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on standardized, large-scale *resource sharing*. Grid systems can be thought of in analogy to the power grid: Just as a power grid provides us with power to our homes and businesses, a Grid system provides us with computational resources on demand. In the same utility-fashion, a Grid system is capable of integrating an arbitrary number of computing devices, adding to the computing capabilities and problem resolution task.

A prominent definition of the Grid originates from Foster [FKT01], which states that the Grid is about "flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources." At the heart of Grid is the concept of virtual organization. A virtual organization is a dynamic collection of individuals, institutions and resources bundled together in order to share resources as they tackle common goals. This resource sharing is not primarily file exchange, but rather direct, controlled access to computing resources (CPU cycles, disk storage, data, software, peripherals), as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science and engineering.

While the application domains for the Grid are manifold, the Grid has gained particular popularity in a domain called eScience, which subsumes a range of applications in a variety of scientific disciplines including high-energy physics, global climate change, and life sciences. Common to these application domains is that they produce immense amounts of data, which need to be accessed and analyzed by communities of researchers that are typically geographically distributed. Further, the application domains of eScience evolve very quickly, and so does the terminology of the field.

While traditionally many Grid systems concentrate on the utilization of distributed *computational resources*, *informational resources* are of paramount importance. In fact, an entire class of Grid systems, called Data Grids [CFK+01], addresses the topic of large amounts of data in Grid environments. Initially, the focus of Data Grids was on providing

scalable and performant storage systems for managing large data sets, including aspects such as high-speed data movement, scalable query processing, etc. However, storage systems constitute just one aspect of Data Grids: The other important aspect is that of managing the data about data that is stored in storage systems, i.e. the metadata management. In modern Data Grid architectures we find data storage and metadata as the fundamental services.

The importance of metadata management in Grid systems is a direct implication of the core goal of the Grid, i.e. coordinated resource sharing. There are a number of problems associated with resource sharing among a set of individuals or groups:

- *Integration.* A Grid involves a multiplicity of resources that are heterogeneous and might span multiple administrative domains. It thus requires the ability to integrate multiple distributed, heterogeneous, and independently managed data sources.

- *Resource Discovery.* Grid applications require that Grid systems should enable new capabilities to be constructed dynamically and transparently from distributed services. In order to engineer new Grid applications it is desirable to be able to reuse existing components and information resources and to assemble and coordinate these components in a flexible manner. This requires the ability to provide resource discovery mechanisms, which allow the user to find resources based on their characteristics.

- *Decentralization.* The nature of the Grid prohibits building systems with a centralized control. Instead one must provide the necessary infrastructure for coordination among the resources that reflects the autonomy of entities in the virtual organization. This also includes that in a Grid the availability of resources may be limited, and resource failures may be the rule rather than exception.

Further, open standard protocols and frameworks are essential for the realization of the Grid. The use of open standards provides interoperability and integration facilities. These standards must be applied for resource discovery, resource access and resource coordination. In order to achieve true interoperability, recent initiatives aim at aligning Grid technologies with other computing technologies that address similar problems. As a consequence, the Grid has moved away from a collection of protocols and is converging with service-oriented architectures (SOA) [JEF04].

Example Systems. There is now a growing number of existing Grid-related projects, ranging from infrastructure, such as Globus [FK97], OGSA [FKNT02], to specific applications and systems, such as DataGrid [HJMS+00].

The Open Grid Services Architecture (OGSA) is the result of the alignment of existing Grid standards with emerging service-oriented architectures; it unites Web Services

with Grid requirements and techniques. OGSA provides a uniform way to describe Grid services and define a common pattern of behavior for these services.

In 2002, IBM launched the *On Demand* initiative [Fel04], which relies on Grid technology as one of its core technical pillars. There are three essential capabilities that support the fundamental concept of an On Demand Business: (1) *Integration* based on service-oriented architectures, realized with Web services and open standards, (2) *Virtualization* based on Grid computing that allows distributed computing resources to be shared and managed as if they were a single, large, virtual computer, and (3) *Autonomic Computing*, a set of technologies with the goal to enable computer systems ability to manage, repair, and protect themselves. It is expected that initiatives such as the On Demand initiative will expedite the advance of Grid technologies in industry.

2.5 Characteristics of Distributed Information Systems

In the previous sections we have discussed three classes of distributed information systems: federated databases, Peer-to-Peer systems, and Grid systems. While each of them pursues specific goals and puts emphasis on different aspects, they all share common characteristics that pose particular challenges that need to be addressed when realizing such systems. These challenging characteristics can be classified along the following three dimensions: (1) the *heterogeneity* of nodes, i.e. discrepancies in the way the data is modeled and represented, (2) *dynamics*, i.e. changes in the structure of the system as well as changes in the information at the individual nodes, and (3) *autonomy*, i.e. the ability of nodes to take decisions independently, which poses additional requirements on the coordination models to achieve scalability. In the following subsections we discuss each of the dimensions in more detail.

2.5.1 Heterogeneity

In distributed information systems we are faced with a multitude of data sources of heterogeneous nature. This heterogeneity is a major barrier to interoperability between resources. Heterogeneity refers to discrepancies in the way that data is modeled, represented, and interpreted. Here we can distinguish heterogeneity on three levels: the *data model level*, the *schema level*, and the *data level*.

- The first discrepancy at the level of the data model occurs if nodes rely on different underlying models to describe their data structures and constraints. For example, one node may rely on the relational data model, while another node uses a hierarchical data model.

- The second form of heterogeneity at the schema level refers to differences in the way how a domain is modeled. Here we can further distinguish between *struc-*

tural and *semantic* differences. Structural heterogeneity exists if elements have the same meaning, are modeled with the same data model, but structured in a different way, e.g. if attributes are grouped into different tables. Semantic heterogeneity occurs when there is disagreement about the intended meaning, interpretation, or intended use of the data. Semantic heterogeneity is especially a challenge, since the schemas used to describe the data often do not provide enough semantics to interpret the data consistently. Orthogonal to the distinction between structural and semantic differences is the problem that the schemas of two nodes may model different domains that share only limited overlap, but still these correspondences need to be related.

- The third form of heterogeneity at the data level may occur if data values are represented differently. For example, the data value of a temperature may be given in Celsius or in Fahrenheit. Heterogeneity on the data level often arises due to missing conventions for representation, abbreviations, but possibly also syntactic errors.

For readers familiar with Model Driven Architectures (MDA), it may be noted that the above levels – data model-, schema-, and data level – directly correspond to the metalevels M2 (metamodel), M1 (model), and M0 (data) of the Meta Object Facility (MOF) [Gro03]. In the literature we find many other possible classifications of heterogeneity on different levels of detail. For example, [SvH05] distinguishes the three categories of syntax, structure, and semantics.

Of course, there may also be heterogeneity on additional dimensions that require integration on a system, process and application level. However, these aspects are beyond the focus of this work.

2.5.2 Dynamics

The second common characteristic shared by distributed information systems is that of dynamics: distributed information systems are subject to continuous change and evolution. Changes may occur for a number of different reasons. For an in-depth discussion of these reasons we recall the definition of an information system: An information system is a system that manages facts – or data – about some domain for a specific purpose. Thus it may be that:

- The data that is managed in the system changes, as new facts are added, deleted or modified.

- The domain itself about which data is managed undergoes changes. Such changes are rather natural, as the domain refers to some aspect of the real world, which is constantly changing. Often such changes in the domain require changes to the data structure, e.g. the schema, used to represent the data.

- The purpose for which the data is managed changes, or in other words, the intended use of the data changes. This may happen if the application environment or the requirements of the entity that makes use of the data change after the distributed information system has been built.

The first type of change is well supported by data management systems, which are able to efficiently handle updates to the data. The other two types require changes to the interpretation of the data and are in this sense considerably harder and typically not well supported.

An additional aspect of change is introduced by the distribution of the data. In a distributed information system the data is physically distributed across multiple nodes, but is logically related. This distribution and these relations may be dynamic. As a consequence, the structure, i.e. the topology of the system, may change. New nodes may join or leave the system, new data sources may appear. In a decentralized system there may not even be control over this process.

Capturing changes can occur either from explicit requirements or from the result of change discovery methods, which induce changes from patterns in data and usage. Explicit requirements are generated, for example, by a user who wants to adapt the information system to new requirements or by the end-users who provide the explicit feedback about the usability of the information system. The changes resulting from such requirements are called top-down changes. Implicit requirements leading to bottom-up changes are reflected in the behavior of the system and can be discovered only through the analysis of this behavior.

A very important problem in change management is that of maintaining consistency in order to allow a meaningful interpretation of the data. The consistent management of changes is already a challenge in the absence of distribution, as it needs to be ensured that the changes are performed in a way consistent with given constraints of integrity, e.g. defined by a schema. In the presence of distribution, changes will need to be propagated across nodes to ensure that the interoperability is maintained in a consistent manner.

2.5.3 Autonomy

Autonomy refers to the fact that nodes can take decisions independently. It is thus closely related to the degree of decentralization. In the case of completely centralized topologies, one node has complete control over all other nodes, whereas in completely decentralized topologies, there is no central control and all nodes act autonomously.

Autonomy of nodes is a key property of distributed information systems. As we have seen in the discussion of the three classes of distributed information systems, it is a desirable property for a number of reasons, which we can summarize along the following dimensions according to [GMK88]:

- *Organizational Issues.* In a distributed information system used by a large organization, node autonomy is a natural extension of departmental autonomy. In a virtual organization, a similar situation exists where participants are in control of different nodes. Here, the degree of required autonomy may be even higher.
- *Diversity of Local Needs.* Having autonomous nodes, the system can be more easily tailored to the needs of local users. This is an important aspect as many distributed information systems are characterized by a diversity of functionality and performance requirements.
- *Data Security.* Autonomy further allows local control of the nodes over the data, which is essential in systems sensitive to unauthorized data access.
- *Failure Tolerance.* Node autonomy supports the containment of the effects of a local failure at a given node in the system: Independent nodes can continue to function despite such a failure.
- *Lower Costs.* The ability of nodes to perform local operations autonomously allows reducing the number of messages to be sent between nodes and thus decreasing the cost of operation.

To gain a better understanding of the concept of autonomy it is useful to classify it into several types, each one covering a different aspect. [VPZ88] distinguishes three types of autonomy:

- *Design autonomy*, which refers to the fact that nodes choose their own design (for example with respect to data being managed, data models, functionality and implementation),
- *Communication autonomy*, which refers to the ability to decide what information to provide to other components of the system and to decide whether to communicate with other nodes, and
- *Execution autonomy*, which refers to the ability to execute local operations without external interference from other nodes.

Of course, autonomy does not come for free. In particular, the autonomy of nodes poses special requirements on the coordination models of the system: While in centralized systems we may assume one (or many) nodes to centrally coordinate the system, this assumption does not hold in decentralized systems and thus requires new coordination models that scale in the presence of autonomous nodes and in the absence of centralized control.

According to the different types of autonomy, we can also distinguish corresponding aspects of coordination: Existing approaches of distributed information access almost

always assume a setting where information providers agree on some global model or schema that is used to access the information. In a decentralized setting with autonomous nodes, this assumption does not longer hold. Therefore, one has to find a way to deal with the existence of multiple, distributed and frequently changing views on the domain. What is required are models for meaning coordination that operate without centralized control.

Communication autonomy means that nodes are free to decide what information to provide and which other nodes to communicate with. Again, this decision is a purely local one without centralized control. This has consequences for the network organization and the coordination to direct requests to the nodes able to respond to the request.

Execution autonomy requires nodes in a distributed information system to be able to perform local operations without the interference from other nodes. This means that while interoperation with other nodes is desired, there must be no dependence on other nodes. The coordination models for distributed operations such as query processing must take this fact into account. Execution autonomy has a great impact on transaction processing, which is an important problem in distributed data management, but less related to the interpretation of the data and therefore beyond the scope of this work.

Further, we need to take the degree of distribution in terms of the number of nodes into account, which poses additional requirements on the coordination models to achieve scalability.

2.6 Conclusions

In this chapter we have discussed distributed information systems as systems that manage and allow to interpret data that is physically distributed, but logically related. Further we have identified three common characteristics of distributed information system: heterogeneity, dynamics, and the autonomy of nodes. These three characteristics are valid across different kinds of distributed information systems, as we have seen in the analysis of federated databases, Peer-to-Peer, and Grid systems. Similar classifications have been proposed for example in [SL90] with the dimensions of autonomy, heterogeneity, and distribution. As in our work we focus explicitly on distributed information systems, we have not considered distribution as a separate characteristic. The aspect of dynamics has been included especially in more recent classifications, e.g. [BKLW99] has introduced *evolvability* as additional characteristic.

In the next chapter we present an overview of how ontologies can be used to address the challenges arising from the heterogeneity, dynamics, and autonomy in distributed information systems. In the subsequent parts of this book we develop specific semantic technologies for solving particular problems in distributed information systems.

Chapter 3

Ontologies

In this chapter we introduce the reader to the ontology model that will serve as foundation throughout this work. In Section 3.1 we provide a general introduction to ontologies as a means to explicitly and formally specify conceptual models with well-defined semantics. In Section 3.2 we present the particular ontology language which we base our work on: the Web Ontology Language OWL. Specifically, we rely on OWL DL, a particular species of OWL that is grounded in an expressive, well-understood description logic.

In order to access the information represented using an ontology, we require a query language. Thus far, no query languages for OWL have been standardized. Several proposals for such query languages exist. While they differ in detail, all of them are able to express conjunctive queries, which have been found useful as query language in practice. In Section 3.3 we introduce conjunctive queries as a formalism to query description logic ontologies. We also discuss SPARQL, a proposal for a Semantic Web query language, as a means to express such conjunctive queries.

While OWL is a very expressive ontology language by itself, there are certain tasks that cannot be accomplished within description logics. Rules are of particular importance, especially in the context of information integration. In Section 3.4, we present a brief overview of so called DL-safe rules, a decidable fragment of the Semantic Web Rule Language (SWRL).

3.1 A Short Introduction to Ontologies

The word "ontology" comes from philosophy, where it is used to describe the existence of being. It was first introduced by Aristotle in his metaphysics [Ari08]. The term has been taken up by the AI community, where ontologies have been studied since the early 1990s in areas such as knowledge engineering [SBF98], natural language processing [NR04], and knowledge representation [DFvH+00]. Various definitions have been pro-

posed to assign to the term "ontology" a computational meaning. The most prominent one stems from Gruber [Gru93]: "An ontology is an explicit specification of a conceptualization." [Bor97] extended the definition, requiring the specification to be formal and the conceptualization to be shared. According to Studer [SBF98], a *conceptualization* refers to an abstract model of some phenomenon, *explicit* means that the concepts and the constraints on their use are explicitly defined, *formal* refers to the fact that the ontology should be machine-readable, and *shared* reflects the notion that an ontology captures consensual knowledge.

As such, ontologies provide a shared and common understanding of a domain, an understanding that can be communicated across people and application systems. These promises are the reason for the recent spread and success of ontologies in application areas such as intelligent information integration, cooperative information systems, information retrieval, electronic commerce, Semantic Web Services and knowledge management.

With such wide spread application areas and a definition of an ontology as generic as the one by Gruber [Gru93], it is not surprising that nowadays a wide range of models are called ontology. Instead of restricting the definition of an ontology at this point and thus excluding particular models from being considered an ontology, we rather present a classification of different types of ontologies.

Classification of Ontologies We introduce a classification that has been inspired by [Obe05], and [Gua98]. Ontologies can be classified along two dimensions:

- the level of *formality* and
- the level of *generality*.

According to the level of formality, we can distinguish the following:

- An *informal ontology* is the simplest type that comprises a set of concept names organized in a hierarchy, possibly with definitions in natural language.
- A *formal ontology* further includes axioms and definitions stated in a formal language.

A similar classification is given by Gruber [Gru04], where he additionally distinguishes *semi-formal* ontologies. A semi-formal ontology in this definition is a partially formal, but largely informal ontology, which still allows useful computations on the formal part.

According to the level of generality, ontologies can be classified into the following types:

- A *top-level ontology* defines very general concepts such as space, time, object, event, etc., which are independent of a particular domain. Commonly, top-level

ontologies are also called upper level or foundational ontologies. Popular examples include DOLCE [GGM+02] and SUMO [NP01].

- A *core ontology* is more specific than a top-level ontology, but still defines concepts which are common across various domains. An example is the Core Ontology of Services, which specializes the DOLCE ontology to model aspects related to the management of services [Obe05]. A core ontology can provide the basis for specialization into task- or domain-specific concepts.

- A *domain ontology* defines concepts associated with a specific domain. For example, the SWRC ontology [SBH+05], which we will use as a running example throughout this work, models the domain of a research community.

- A *task ontology* defines concepts related to the execution of a particular task or activity, e.g. planning, scheduling, or problem solving [vHSW97].

- An *application ontology* defines concepts essential for a particular application, often depending both on a particular domain and task.

A further classification of ontologies would be possible along the knowledge representation formalism and the corresponding inference mechanisms used or provided by these formalisms. Common to all ontology languages is that the conceptualization takes the form of a definition of the properties of important concepts and relationships. We refrain from providing an introduction to all of these formalisms, and instead refer the reader to [SS04] for an overview of ontology representation formalisms and languages.

3.2 The Family of OWL Languages

Traditionally, a number of different knowledge representation paradigms have competed to provide languages for representing ontologies, including most notably description logics and frame logics. With the advent of the OWL Web Ontology Language, developed by the Web Ontology Working Group and recommended by World Wide Web Consortium (W3C), a standard for the representation of ontologies has been created. Adhering to this standard, we base our work on the OWL language (in particular OWL DL, as discussed below) and describe the developed formalisms in its terms.

In this section we introduce the OWL ontology language and description logics as its logical foundations. This introduction is to a great extent inspired by [HPSvH03] and [BHS04]. The OWL ontology language is based on a family of description logics languages. Within this family, three members have been standardized as sublanguages of OWL: OWL Full, OWL DL, and OWL Lite. These sublanguages differ in expressiveness, i.e. in their provided constructs and the allowed combinations of these constructs.

- OWL Full is the most expressive of the members of the OWL family. It has mainly been defined for compatibility with existing standards such as RDF (Resource Description Framework, [MM04]). OWL Full is undecidable and thus impractical for applications that require complete reasoning procedures.
- OWL DL is a sublanguage that was designed to regain computational efficiency. It directly corresponds to the $\mathcal{SHOIN}(\mathbf{D})$ description logic, a decidable logic with NEXPTIME complexity. For $\mathcal{SHOIN}(\mathbf{D})$, practical reasoning algorithms are known, and increasingly more tools support this or slightly less expressive languages.
- OWL Lite corresponds to the less expressive $\mathcal{SHIF}(\mathbf{D})$ description logic. Its complexity is known to be EXPTIME, which means that reasoning with OWL Lite is still intractable. However, the restricted expressiveness makes OWL Lite conceptually easier to grasp and thus interesting for applications that require a light-weight ontology language.

The respective species are subspecies of one another, which means that every OWL Lite ontology is an OWL DL ontology, and every OWL DL ontology is an OWL Full ontology. Thus, if in the following we rely on OWL DL as ontology language, this includes all subspecies of OWL DL as well. In addition to these species, further relevant subspecies have been identified. For example, OWL DLP is a proper subspecies of OWL Lite [GHVD03, HHK+05]. Although it is only marginally less expressive than OWL Lite, it enjoys polynomial complexity.

Several different syntaxes for OWL DL have been defined, including an abstract syntax as well as XML and RDF-based syntaxes. For our presentation, we use the traditional description logic notation since it is more compact. For the correspondence between this notation and various OWL DL syntaxes, see [HPS04b].

Before we discuss the description logic underlying OWL DL in detail, we provide a brief overview of description logics in general.

3.2.1 OWL as Description Logics

Description logics are a family of class-based knowledge representation formalisms. In description logics, the important notions of a domain are described by concept descriptions that are built from concepts (unary predicates) and roles (binary predicates) by the use of various concept and role constructors. In addition to these concept descriptions, it is possible to state facts about the domain in the form of axioms. Terminological axioms make statements about how concepts or roles are related to each other, assertional facts make statements about the properties of individuals of the domain.

In the design of description logics, emphasis is put on retaining decidability of key reasoning problems and the provision of sound and complete reasoning algorithms. As

the name suggests, Description Logics are logics, i.e. they are formal logics with well-defined semantics. Typically, the semantics of a description logics is specified via *model theoretic semantics*, which explicates the relationship between the language syntax and the models of a domain.

An interpretation consists of a domain and an interpretation function, which maps from individuals, concepts and roles to elements of the domain, subsets of the domain and binary relations on the domain, respectively. A description logic knowledge base consists of a set of axioms, which act as constraints on the interpretations. The meaning of a knowledge base derives from features and relationships that are common in all possible interpretations. An interpretation is said *to satisfy a knowledge base*, if it satisfies each axiom in the knowledge base. If there are no possible interpretations, the knowledge base is said to be *inconsistent*. If the relationship specified by an axiom holds in all possible interpretations of a knowledge base, the axiom is said to be *entailed* by the knowledge base.

Before we formally define the syntax and semantics of the description logics underlying OWL, we here informally introduce the language constructs of the description logics $\mathcal{SHOIN}$. In particular we can build complex classes from atomic ones using the following constructors:

- $C \sqcap D$ (intersection), denoting the concept of individuals that belong to both C and D,
- $C \sqcup D$ (union), denoting the concept of individuals that belong to either C or D,
- $\neg C$ (complement), denoting the concept of individuals that do not belong to C,
- $\forall R.C$ (universal restriction), denoting the concept of individuals that are related via the role R only with individuals belonging to the concept C,
- $\exists R.C$ (existential restriction), denoting the concept of individuals that are related via the role R with some individual belonging to the concept C,
- $\geq n\, R$, $\leq n\, R$ (qualified number restriction), denoting the concept of individuals that are related with at least (at most) n individuals via the role R.
- $\{c_1, \ldots, c_n\}$ (enumeration), denoting the concept of individuals explicitly enumerated.

Based on these class descriptions, axioms of the following types can be formed:

- concept inclusion axioms $C \sqsubseteq D$, stating that the concept C is a subconcept of the concept D,
- transitivity axioms $\mathsf{Trans}(R)$, stating that the role R is transitive,

- role inclusion axioms $R \sqsubseteq S$ stating that the role R is a subrole of the role S,
- concept assertions $C(a)$ stating that the individual a is in the extension of the concept C,
- role assertions $R(a, b)$ stating that the individuals a, b are in the extension of the role R,
- individual (in)equalities $a \approx b$, and $a \not\approx b$, respectively, stating that a and b denote the same (different) individuals.

Datatypes In addition to the "abstract" classes considered so far, the $\mathcal{SHOIN}(\mathbf{D})$ description logic further supports reasoning with concrete datatypes, such as strings or integers. For example, it is possible to define a minor as a person whose age is less than or equal to 18 in the following way: $\texttt{Minor} \equiv \texttt{Person} \sqcap \exists \texttt{age}.\leq_{18}$. The interpretation of datatypes and values in $\mathcal{SHOIN}(\mathbf{D})$ includes an additional interpretation domain for data values (concrete domain), which is disjoint from the domain of individuals. Further, roles are divided into abstract roles, which are interpreted as binary relations on the domain of individuals and concrete roles which are interpreted as binary relations between the domain of individuals and the concrete domain.

Example 1 *In this example we build a small terminology about the bibliographic domain (based on a fragment of the SWRC ontology [SBH+05]. Consider the atomic concepts* `Person`, `Publication`, `Article` *and* `Book`, *as well as the abstract role* `author` *and the concrete role* `title`. *We can build a terminology stating that*

- $\texttt{Article} \sqsubseteq \texttt{Publication}$, *every article is a publication,*
- $\texttt{Book} \sqsubseteq \texttt{Publication}$, *every book is a publication,*
- $\texttt{Article} \sqsubseteq \neg\texttt{Book}$, *articles and books are disjoint, and*
- $\texttt{Publication} \sqsubseteq \forall \texttt{author}.\texttt{Person}$, *the author of every publication is a person.*

Further we can assert facts about individuals, `codd` *and* `relational_model`, *stating that*

- $\texttt{Article}(\texttt{relational\_model})$, *the individual* `relational_model` *belongs to the concept* `Article`,
- $\texttt{title}(\texttt{relational\_model}$, *"The Capabilities of Relational Database Management Systems"*$)$, *the title of* `relational_model`,
- $\texttt{Person}(\texttt{codd})$, *the individual* `codd` *belongs to the concept* `Person`,

- $\texttt{author}(\texttt{relational\_model}, \texttt{codd})$, *codd is the author of* `relational_model`,
- `relational_model` $\not\approx$ `codd`, *the individuals* `codd` *and* `relational_model` *are two different individuals*[1].

3.2.2 OWL as a "Web" Ontology Language

There are several features of the OWL language that distinguish it from pure classical description logic, but make it a true ontology language for the web. In fact, these additional characteristics make OWL practically applicable not only for the web, but for open distributed information systems in general.

The main characteristic to be mentioned is the tight integration with existing standards in general, and web standards in particular. These standards include the use of the namespace concept to organize its vocabulary and the use of URI references as names to uniquely identify ontology elements. Last not least, several XML-based syntaxes for OWL have been defined to enable an easy interchange of ontologies. OWL also reuses the facilities of XML Schema [BM04] to provide datatypes and data values.

The following example shows a fragment of the SWRC ontology in OWL/RDF syntax that demonstrates the use of some of the above mentioned features:

```
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
    <owl:Ontology rdf:about="http://swrc.ontoware.org/ontology#"/>
    <owl:Class rdf:about="http://swrc.ontoware.org/ontology#Article">
        <rdfs:label>Article</rdfs:label>
        <rdfs:subClassOf>
            <owl:Class
            rdf:about="http://swrc.ontoware.org/ontology#Publication"/>
        </rdfs:subClassOf>
    ...
    <owl:Class
...
</rdf:RDF>
```

Additionally, for OWL DL and OWL Lite, an abstract syntax has been defined, which due to its frame-like compact representation is most suitable for consumption by humans unfamiliar with the classical DL-based syntax.

Furthermore, OWL allows to attach annotations to ontology elements. Even though these annotations do not carry a logical meaning, they are important

[1]Please note that this fact is not trivial, as OWL does not take the unique names assumption.

for representing human readable descriptions (e.g. `rdfs:label`), versioning information (e.g. `owl:versionInfo`), etc. Finally, ontology properties allow to describe the relationships with other ontologies, namely `owl:imports`, `owl:backwardCompatibleWith`, `owl:incompatibleWith`, and `priorVersion`. These ontology properties do not carry a logical meaning, with the exception of `owl:imports`, where the meaning of the imported ontology is considered to be part of the meaning of the importing ontology. The `owl:imports` primitive thus provides an effective – albeit simple – means for modularization.

3.3 Query Languages

Query languages are an important tool in information management. They provide the means to access data that is managed using a particular data model. Surprisingly, despite the advance of OWL as an ontology language, the work on adequate query languages for OWL is still in its infancy. This is partly due to the fact that traditionally the focus of Description Logics has not been on managing large sets of assertional facts, but rather on reasoning over the terminological knowledge. Recently, there have proposals for query answering over description logics that are based on a reduction to standard description logic reasoning problems. While this is theoretically elegant, it is impractical, as effectively for every individual in the knowledge base one needs to check whether it satisfies the query. On the other hand, there is a large body of work on query languages in database management. Some attempts have been made to build on this work in the context of query languages for RDF.

In [HBEV04] we have presented a comparative analysis for six RDF query languages. While some of these query languages are used in many systems as OWL query languages (simply treating OWL as an extension of RDF), such an approach has major drawbacks. The main problem lies in the fact that the semantics of RDF query languages is typically defined via a pattern matching over a particular RDF graph or a set of RDF triples, which corresponds to *one* particular model of the RDF knowledge base. Such a semantics is incompatible with that of description logics in the sense that an answer over a description logic knowledge base is supposed to hold in *any* model, of which there may be infinitely many.

A query language of particular importance is SPARQL, which is currently being standardized by the W3C as the standard query language for the Semantic Web. While SPARQL in principle follows the same pattern matching approach as other RDF query languages, in essence it only standardizes the *syntax* of a query language, but leaves the *semantics* up to the implementation. Although the lack of a well-defined semantics can be seen as a deficiency, it can also be seen as a feature, as it allows us to use SPARQL merely as a syntax carrier and to separately assign it a well-defined meaning based on grounding in descriptions logics. In the following we thus present the semantics of

conjunctive queries over OWL DL ontologies and afterwards show how SPARQL can be used to encode such conjunctive queries. In our discussion we will further analyze how these formalisms meet generic criteria desired for any query language. We here review these criteria from [HBEV04]:

- *Expressiveness.* Expressiveness indicates how powerful queries can be formulated in a given language. Typically, a language should at least provide the means offered by relational algebra, i.e. be relationally complete. Usually, expressiveness is restricted to maintain other properties such as safety and to allow an efficient (and optimizable) execution of queries.
- *Closure.* The closure property requires that the results of an operation are again elements of the data model. This means that if a query language operates on the graph data model, the query results would again have to be graphs.
- *Adequacy.* A query language is called adequate if it uses all concepts of the underlying data model. This property therefore complements the closure property: For the closure, a query result must not be outside the data model, whereas for adequacy the entire data model needs to be exploited.
- *Orthogonality.* The orthogonality of a query language requires that all operations may be used independently of the usage context.
- *Safety.* A query language is considered safe, if every query that is syntactically correct returns a finite set of results (on a finite data set). Typical concepts that cause query languages to be unsafe are recursion, negation and built-in functions.

3.3.1 Conjunctive Queries

Conjunctive queries are a popular formalism, well explored in database theory, capable of expressing the class of selection/projection/join/renaming relational queries. The vast majority of query languages for many data models used in practice fall into this fragment. Because conjunctive queries have been found useful in diverse practical applications, it is natural to use them as an expressive formalism for querying ontologies.

When talking about conjunctive queries, we need to distinguish between the query as a syntactic object and the *query mapping*, i.e. the function defined by a query under a specific semantics. Informally, a *conjunctive query* $Q(\mathbf{x}, \mathbf{y})$, is a conjunction of DL-atoms, i.e. atoms over DL-concepts and roles. $\mathbf{x}$ and $\mathbf{y}$ denote sets of *distinguished* and *non-distinguished* variables, respectively. Intuitively, the semantics of a conjunctive query is to ask for concrete individuals that are valid fillers for the distinguished variables, with the non-distinguished variables existentially bound. A more formal definition of the semantics of conjunctive queries over $\mathcal{SHOIN}(\mathbf{D})$ ontologies is given in Chapter 6.2.

Example 2 *The following conjunctive query asks for the title of articles written by the individual codd:*

$Q(t, x) : \textit{Publication}(x) \wedge \textit{title}(x, t) \wedge author(x, codd)$

3.3.2 SPARQL

SPARQL was originally designed as RDF query language. As every OWL ontology can be encoded in an RDF graph, SPARQL can serve – at least syntactically – as an OWL query language.

The SPARQL query language is based on matching graph patterns. The simplest graph pattern is the triple pattern. In terms of OWL, every unary assertion $C(a)$ can be represented as a triple $(a\ \textit{rdf:type}\ C)$ and every role assertion $R(a, b)$ can be represented as a triple $(a\ R\ b)$. Triple patterns in queries additionally may contain variables in the subject, predicate or object positions. Combining triples gives a basic graph pattern, where an exact match to a graph is needed to fulfill a pattern.

Queries are composed of two main building blocks, the `SELECT` or `CONSTRUCT` clause and the `WHERE` clause. The `SELECT` or `CONSTRUCT` clause identifies the variables to appear in the query results, they thus correspond to the distinguished variables of a conjunctive query. The `WHERE` clause identifies the triple pattern to match. Let us look at an example:

```
PREFIX  swrc: <http://swrc.ontoware.org/ontology#>

SELECT ?book WHERE {
  ?book rdf:type swrc:Book
}
```

The `PREFIX` clause simply allows to declare namespace abbreviations, in this case for the SWRC ontology. The `SELECT` clause specifies that the `?book` variable should be returned. An optional `FILTER` clause further allows to filter variable bindings, as in the following query that restricts the bindings of books to those that were published after the year 2000:

```
PREFIX  swrc: <http://swrc.ontoware.org/ontology#>

CONSTRUCT { ?book swrc:year ?year } WHERE {
  ?book rdf:type swrc:Book .
  ?book swrc:year ?year . FILTER (?year > 2000)
}
```

This query also demonstrates the use of the `CONSTRUCT` clause, which unlike the `SELECT` queries does not return tuples of variable bindings, but allows to specify triples that should be returned.

Multiple patterns can be joined with the `UNION` clause, as in the following query that asks for all publications that are either a book or a part of a book:

```
PREFIX  swrc: <http://swrc.ontoware.org/ontology#>

SELECT ?x WHERE {
  ?x rdf:type swrc:Book
  UNION
  ?x rdf:type swrc:InBook
}
```

Expressiveness. In terms of expressiveness, an important criterion is relational completeness, which requires that certain basic algebraic operations are supported, i.e. (i) selection, (ii) projection, (iii) cartesian product, (iv) difference and (v) union. These operations can be combined to express other operations such as set intersection, several forms of joins, etc. SPARQL is not relationally complete. It does support the relational operations of selection (`FILTER` clause), projection (`SELECT` clause) and cartesian product (combination of triple patterns in the `WHERE` clause) as well as union (joining patterns with the `UNION` clause). However, it does not support difference, as there is no notion of negation in SPARQL. For example, it is thus not possible to ask for all publications that are not books.

Closure. The SPARQL language is closed, if one considers `CONSTRUCT` queries: The queries operate in triples, and also return triples as results. This is an important feature that allows to compose queries on the one hand, and to directly exchange query results as ontologies on the other hand.

Adequacy. As mentioned above, SPARQL is only partially adequate for querying OWL DL ontologies: SPARQL supports all elements of the OWL DL data model that are relevant for querying, but it also provides many features, which have not been presented above, that are not applicable to OWL DL ontologies (e.g. reification).

Orthogonality. Unfortunately, SPARQL is orthogonal only to a limited extent, as most of the operators have a very narrow scope of usage. For example, the `FILTER` operator can only be used inside of a `WHERE` clause.

Safety. The safety of the language is obviously obviously depends on the semantics that is given to it. So far, SPARQL only defines the syntax of a query language. For the semantics of conjunctive queries over description logics knowledge bases, a sound and complete decision procedure exists, which implies safety.

Concluding, one can say that SPARQL at this time is an acceptable compromise for querying OWL ontologies, provided that is used as a syntax carrier for conjunctive queries, whose semantics is adequately defined separately using model theory.

3.4 Rule Languages for OWL

Although the description logics underlying OWL are very expressive ontology languages, they cannot be used for modeling particular forms of axioms. Specifically, description logic axioms are restricted to a tree-structure that disallows to model rule-like

axioms commonly needed for complex data integration tasks that involve queries, views and transformations. This situation has lead to several proposals for the combination of ontology languages such as OWL with rule-based languages, which are currently controversially discussed [W3C05]. Just recently the W3C has chartered a working group for the definition of a standardized Rule Interchange Format (RIF)[2]. The Semantic Web Rule Language (SWRL, [HPS04a]) is currently the most prominent proposal for an extension of OWL DL with rules. From the Description Logics perspective, a main goal in the definition of a rule language for OWL is to retain decidability of the main reasoning problems. Unfortunately, SWRL is known to be undecidable. Recently, DL-safe rules, which can be seen as a syntactic fragment of SWRL, have been proposed as a decidable rule extension.

3.4.1 SWRL - Semantic Web Rule Language

SWRL allows the use of datalog-like horn rules together with OWL axioms. These rules are of the form of an implication between an antecedent (body) and consequent (head). The intuitive meaning of a rule is that whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. Both the antecedent (body) and consequent (head) of a rule consist of zero or more atoms. Atoms can be of the form $C(x)$, $R(x, y)$, $x \approx y$ or $x \not\approx y$, where C is an OWL DL concept, R is a role, and x,y are either variables, individuals or data values.

Example 3 *The fact that a person is an expert on a topic if he has authored a publication on that topic, can be asserted with the following rule:*

$$expertOn(z, y) \leftarrow Publication(x) \wedge isAbout(x, y) \wedge author(x, z)$$

A great advantage of SWRL is the tight integration with the existing OWL standards: A high-level abstract syntax is provided that directly extends the OWL abstract syntax. Further, an XML based syntax has been defined An extension of the OWL model-theoretic semantics provides a formal meaning for SWRL ontologies.

Example 4 *The rule from Example 3 stating that a person is an expert on a topic if he has authored a publication on that topic, can be expressed in SWRL syntax in the following way:*

```
<swrlx:Ontology swrlx:name=""
    xmlns:owlx="http://www.w3.org/2003/05/owl-xml#"
    xmlns:ruleml="http://www.w3.org/2003/11/ruleml#"
    xmlns:swrlx="http://www.w3.org/2003/11/swrlx#">

<ruleml:imp>
```

[2] http://www.w3.org/2005/rules/wg/charter

```
    <ruleml:_head>
       <swrlx:individualPropertyAtom  swrlx:property="expertOn">
           <ruleml:var>Z</ruleml:var>
           <ruleml:var>Y</ruleml:var>
       </swrlx:individualPropertyAtom>
    </ruleml:_head>
    <ruleml:_body>
       <swrlx:classAtom>
        <owlx:Class owlx:name="Publication" />
        <ruleml:var>X</ruleml:var>
       </swrlx:classAtom>
        <swrlx:individualPropertyAtom swrlx:property="isAbout">
            <ruleml:var>X</ruleml:var>
            <ruleml:var>Y</ruleml:var>
        </swrlx:individualPropertyAtom>
       <swrlx:individualPropertyAtom  swrlx:property="author">
           <ruleml:var>X</ruleml:var>
           <ruleml:var>Z</ruleml:var>
       </swrlx:individualPropertyAtom>
    </ruleml:_body>
</ruleml:imp>
</swrlx:Ontology>
```

3.4.2 DL-safe Rules

An unrestricted combination of description logics with rules leads to undecidability. Intuitively, the undecidability arises because adding rules to description logics causes the loss of any form of tree model property. DL-safe rules [MSS04] have been proposed as a subset of SWRL that allows to regain decidability. In DL-safe rules, the decidability is retained by restricting the interchange of consequences between the component languages, without restricting the component languages themselves. Specifically, concepts (roles) are allowed to occur in both rule bodies and heads as unary (binary) predicates in atoms, but each variable in a rule is required to occur in a body literal whose predicate is neither a DL-concept nor a DL-role. Intuitively, this restriction makes the logic decidable because the rules are applicable only to individuals explicitly introduced in the ABox. In Chapter 6 we discuss the consequences of this restriction in further detail.

3.5 Conclusions

In this chapter we have introduced the reader to ontologies as explicit specifications of a conceptualization. We have presented the OWL ontology language that will serve as the formal model for the work in the subsequent chapters. Further we have discussed query languages for ontologies, including SPARQL, as means to access data represented using

ontologies, as well as rule extensions, including SWRL and DL-safe rules as decidable fragment to increase the expressive power of the OWL ontology language.

In the next chapter we discuss the role of ontologies in distributed information systems and how their application can address the particular challenges of heterogeneity, dynamics and autonomy identified in the previous chapter.

Chapter 4

Ontologies in Distributed Information Systems

In Chapter 2 we have defined and discussed distributed information systems. We have seen that an important aspect is to be able to interpret data from heterogeneous sources in order for the data to be useful. In this chapter we show how ontologies as conceptual models can be used to formalize the semantics of information in distributed information systems.

The specific role of ontologies in distributed information systems is discussed in Section 4.1. In Section 4.2 we analyze how semantic technologies based on ontologies can be used in distributed systems to address the challenges elaborated on in the previous chapter: We discuss *ontology-based information integration* as a solution to address heterogeneity, *ontology evolution* as a possible way to address dynamics, and *ontology-based coordination* to deal with the autonomy of nodes. We also point out which specific aspects of these semantic technologies will be addressed by the contributions presented in the subsequent part of this book. In Sections 4.3–4.5 we show how the specific classes of distributed information systems presented in the previous chapter can benefit from these semantic technologies.

4.1 The Role of Ontologies in Distributed Information Systems

In this section we discuss the role of ontologies in distributed information systems. In order to do so, we come back to our definition of an information system: An information system is a system that manages facts about some domain for a specific purpose. The important aspect here is the purpose, i.e. that either a human or a computer makes use of the data to perform a certain task. This requires an interpretation of the data, relating the data structure with its semantics. In other words, we require a conceptualization as a

structured interpretation of the domain. This conceptualization may be specified using an ontology. The ontology can thus be used to formalize the relationship between the data and its meaning. The advantage of using an ontology for this specification is obvious: The specification is explicit and formal – i.e. provided in an unambiguous language – allowing for a clear interpretation by both humans and machines.

The aspect of providing a *shared understanding* becomes especially important in the context of distribution. In a distributed information system we have data distributed across multiple nodes and possibly different entities making use of the data. The shared understanding is thus essential for a consistent interpretation of the data and interoperability across nodes. However, this does not necessarily mean that all nodes in the distributed information system have to rely on the same global shared ontology. It does allow individual nodes to rely on local ontologies, provided that there are means to relate the local ontology to those of other nodes, as discussed in more detail in Section 4.2.1.

Of course, not only the *domain* of the information system can be specified using an ontology, but also the *application* aspect of the system. Using an application ontology we can specify how the application's functionality is to be implemented and which domain knowledge is required [Bor97]. The benefit of ontologies is that they are well suited for a modular specification of complex structured domain knowledge and that they are independent of the application's implementation language. Sharing and reuse of ontologies across different domains and applications can therefore improve the development of information systems.

A major benefit of the grounding of ontologies in logics is the support for reasoning. Reasoning can be used in different phases of the lifecycle of a distributed information system.

Using ontologies at development time. During the design and specification phase it can be used to check whether the conceptual model is non-contradictory, i.e. free of inconsistencies. Computing the concept hierarchy – to see whether which concepts are specializations or equivalent – helps to assess whether the model has the intended consequences or not [BHS04]. The same holds for the case when multiple ontologies are to be related or change over time.

Using ontologies at runtime. The use of reasoning at runtime, i.e. after the information system is deployed, is of considerably different nature, although pre-computed concept hierarchies can still be useful for many runtime tasks. However, the focus typically is on reasoning tasks such as query answering over integrated data sources or the rewriting of queries.

"A little semantics goes a long way." As the use of reasoning requires the use of formal ontologies, formal ontologies play an important role in distributed information systems. However, there is a very large body of work that can and needs to be done using semi-formal ontologies [SR03]. The reasons lie in the ease with which semi-formal ontologies can be built to a scale that is useful in distributed information systems: First, semi-formal ontologies are typically easier to develop and reuse. The second reason is

related to the trade-off between expressive power and computational complexity: Semi-formal ontologies may be based on limited expressive power, are computationally less expensive and as a consequence often turn out to be more scalable and practically more useful. The observation of the usefulness of less expressive ontology languages also resonates with the so-called Hendler hypothesis [Hen03]: "A little semantics goes a long way." Gruber [Gru04] even claims that "all practical ontologies are semi-formal." Prominent examples, where already informal and semi-formal ontologies can provide benefits for distributed information systems, include exploiting hypernym/synonym relationships in information retrieval, classification of information [BH04], query refinement [Sto04b], presentation and visualization of information [FSvH04], and natural language interfaces to information systems [Cim04].

4.2 Semantic Technologies

In this section we present an overview of how semantic technologies can be used to address the specific challenges we are faced with in distributed information systems. We discuss *ontology-based information integration* as a solution to address heterogeneity, *ontology evolution* as a possible way to address dynamics, and *ontology-based coordination* to deal with the autonomy of nodes.

4.2.1 Ontology-based Information Integration

Dealing with disparate heterogeneous data sources in distributed information sources requires solving many problems of integration. Ontologies have been proposed as a key technology to support integration: A number of disparate data sources can be integrated through a shared understanding of the terminology in their respective ontology.

In Section 2.5.1, we have identified three levels of heterogeneity: the data model level, the schema level, and the data level. In the following we discuss, how ontology-based information integration can serve as solution addressing all three forms of heterogeneity.

Integration on the data model level Regarding the integration on the data model level, the choice of a data model to be used for the integration is an important problem. Ontologies have been proven appropriate as common data model for this task. They have the required expressiveness to capture many data models such as the relational model [BLR03], object-oriented models [FGM00], UML [BCG05], and other models [CLN99]. Further, ontologies are useful to enrich semantics of available schemata, which typically do not provide enough semantics themselves to allow consistent interpretation. Here, ontologies provide a clear semantics by grounding in logics, such that

an unambiguous and precise interpretation of the data is guaranteed. Finally, the grounding in logics enables inference and query support to deduce new facts and more accurate results in response to a given query. The reasoning techniques available in the logical framework provide valuable support for the data modeling activity. For example, they allow to check the consistency of the entire model.

With OWL being approved as a standard for representing ontologies on the Web [HPSvH03], an important step on the road to interoperability has been taken. In its few years of existence, the OWL language has gained popularity across research and industry. In this spirit, the models and methods developed in this work build on the OWL data model.

Integration on the schema level Semantic integration on the schema level involves a number of several steps: (1) identifying correspondences between heterogeneous schemas, (2) representing these correspondences in an appropriate mapping formalism, and (3) using these mappings for a given integration task.

The first important problem in semantic integration is that of discovering semantic correspondences – or mappings – between the vocabularies of different data sources. Mapping discovery is a very active research topic. There exists a multitude of approaches to (semi-)automatically identify correspondences by applying various techniques, including for example linguistic analysis, use of heuristics, machine learning, or graph-based techniques. In the end, most of these techniques rely on a notion of *similarity* to identify corresponding concepts. In Chapter 7 we present a similarity framework for ontologies that allows to combine different notions of similarity relevant for integrating heterogeneous sources.

Once the correspondences between data sources are known, the next important problem is that of representing the mappings using an appropriate formalism and using them for some specific integration tasks, such as data transformation or query answering over heterogeneous data sources. A common approach in data integration is to represent mappings as views over data sources, often using the notion of a global ontology to provide an integrated view over the heterogeneous data sources. If the global ontology is defined as a view over the local sources, we speak of the Global-as-View (GAV) approach, if the local sources are defined as a view over the global ontology, we speak of the Local-As-View (LAV) approach. However, in decentralized distributed information systems, the integration of data sources poses a particular challenge because of the autonomy of the nodes, which often prohibits the definition of a global ontology. Here, direct mappings in both directions between the individual data sources are required. A formalism to represent such mappings for OWL ontologies along with an algorithm for query answering is presented in Chapter 6.

Integration on the data level A similar step as for the mapping discovery on the schema level must be carried out on the data level. Heterogeneity on the data level often arises due to abbreviations, syntactic errors, and missing conventions for representation. We thus need to identify data values that represent the same real-world entity. Popular examples include matching citations of research papers, authors, and institutions. Here again, ontologies can provide the required background knowledge for integration.

Most efforts in semantic integration focus on structured artifacts, but even in the case where no structured data models are used, e.g. for file sharing of unstructured documents, ontologies can help to annotate data sets and to define relationships among heterogeneous data sets.

4.2.2 Ontology Evolution

In distributed information systems, domain knowledge evolves continually. These changes include accounting for the modification in the application domain, incorporating additional functionality according to changes in the user needs, organizing information in a better way, etc. Ontology evolution can be defined as the timely adaptation of an ontology to such changes and the consistent management of these changes [Sto04a]. In [SMMS02] the authors identify a possible six-phase evolution process. As shown in Figure 4.1 the phases are: (1) change capturing, (2) change representation, (3) semantics of change, (4) change propagation, (5) change implementation, and (6) change validation. In the following, we use this evolution process as the basis for an overview of the area. For a more detailed overview we refer the reader to [HVS05].

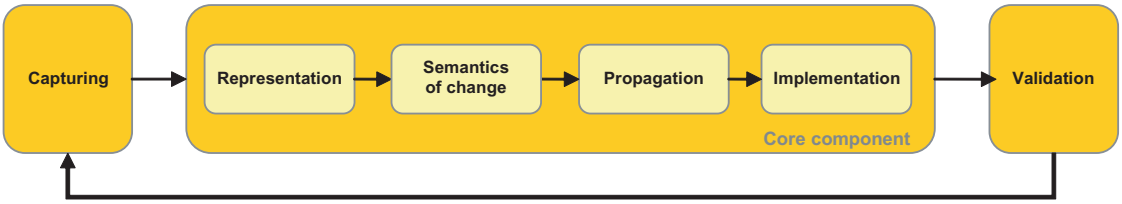


Figure 4.1: Ontology Evolution Process

Change Capturing The process of ontology evolution starts with capturing changes either from explicit requirements or from the result of *change discovery* methods, which induce changes from existing data. Explicit requirements are generated, for example, by ontology engineers who want to adapt the ontology to new requirements or by the end-users who provide the explicit feedback about the usability of ontology entities.

The changes resulting from this kind of requirements are called *top-down* changes. Implicit requirements leading to *bottom-up* changes are reflected in the behavior of the system and can be discovered through the analysis of this behavior. [Sto04a] defines three types of change discovery: structure-driven, usage-driven and data-driven. Data-driven changes are generated by modifications to the underlying dataset, such as text documents or a database, representing the knowledge modeled by an ontology. Whereas structure-driven changes can be deduced from the ontology structure itself, usage-driven changes result from the usage patterns created over a period of time. In Chapter 9 we present a novel technique to capture recommendations for ontology changes from the usage behavior in a community of users based on methods from collaborative filtering.

Change Representation To resolve changes, they have to be identified and represented in a suitable format. That means, the change representation needs to be defined for a given ontology model. Changes can be represented on various levels of granularity, e.g. as elementary or complex changes. A common practice is to provide a taxonomy or ontology of changes for a given ontology model.

In Chapter 8, we define a form of change representation for OWL, where we follow an ontology model influenced by Description Logics, which treats an ontology as a knowledge base consisting of a set of axioms. Accordingly, we allow the atomic change operations of adding and removing axioms. Obviously, representing changes at the level of axioms is very fine-grained. However, based on this minimal set of atomic change operations, it is possible to define more complex, higher-level descriptions of ontology changes. Composite ontology change operations can be expressed as a sequence of atomic ontology change operations. The semantics of the sequence is the chaining of the corresponding functions.

Semantics of Change The semantics of change refers to the effects of the change on the ontology itself and in particular checking and maintaining ontology consistency after the change application. The meaning of consistency much depends on the underlying ontology model. It can for example be defined using a set of constraints, as in the KAON ontology model in [Sto04a], or it can be given a model-theoretic definition. In Chapter 8 we describe the semantics of change for the consistent evolution of OWL ontologies, considering structural, logical, and user-defined consistency conditions. Here, resolution strategies map each consistency condition to a resolution function, which returns for a given ontology and an ontology change operation an additional change operation, which – applied to the ontology – results in an ontology that satisfies the consistency condition. The semantics of OWL ontologies is defined via a model theory: An interpretation satisfies an ontology, if it satisfies each axiom in the ontology. Axioms thus result in semantic conditions on the interpretations. Consequently, contradictory axioms will allow no possible interpretations. The goal of the resolution function is to determine a set of axioms

to remove, in order to obtain a logically consistent ontology with "minimal impact" on the existing ontology. Obviously, the definition of minimal impact may depend on the particular user requirements.

Finally, it should be noted that there exist other approaches to deal with inconsistencies: In Chapter 8.6 we compare consistent evolution of OWL ontologies with other approaches in a framework for dealing with inconsistencies in changing ontologies.

Change Propagation Ontologies often reuse and extend other ontologies. The task of the change propagation phase of the ontology evolution process is to ensure consistency of *dependent artefacts* after an ontology update has been performed. These artefacts may include dependent ontologies, instances, as well as application programs running against the ontology.

[MMS03a] present an approach for evolution in the context of dependent and distributed ontologies. The authors define the notion of Dependent Ontology Consistency and Replication Ontology Consistency for the case of multiple ontologies distributed over multiple nodes. Further, the authors contrast Push-based and Pull-based approaches for the synchronization of dependent ontologies are compared.

Change Implementation The role of the change implementation phase of the ontology evolution process is (1) to inform an ontology engineer about all consequences of a change request, (2) to apply all the (required and derived) changes and (3) to keep track about performed changes.

Change Notification. In order to avoid performing undesired changes, a list of all implications for the ontology and dependent artefacts should be generated and presented to the ontology engineer, who is then able to accept or abort these changes.

Change Application. The application of a change should have transactional properties, i.e. (A) Atomicity, (C) Consistency, (I) Isolation, and (D) Durability. The approach of [Sto04a] realizes this requirement by the strict separation between the request specification and the change implementation. This allows the set of change operations to be easily treated as one atomic transaction, since all the changes are applied at once.

Change Logging. A typical way to keep track of the performed changes is using an evolution log. [Sto04a] proposes an evolution log based on an evolution ontology. The evolution ontology covers the various types of changes, dependencies between changes (causal dependencies as well as ordering), as well as the decision making process.

Change Validation There are numerous circumstances where it can be desirable to reverse the effects of the ontology evolution, as for example in the following cases:

- The ontology engineer may fail to understand the actual effect of the change and approve a change which should not be performed;

- It may be desired to change the ontology for experimental purposes;
- When working on an ontology collaboratively, different ontology engineers may have different ideas about how the ontology should be changed.

It is the task of the change validation phase to recover from these situations. Change validation enables justification of performed changes or undoing them at user's request. Consequently, the usability of the ontology evolution system is increased.

4.2.3 Ontology-based Coordination

As we have seen in Section 2.5.3, the autonomy of nodes in distributed information systems is a prime challenge for the coordination of such systems. Decentralized architectures prohibit to pre-impose top-down coordination models. However, recent advances show that decentralization is not necessarily a threat, but can be exploited as a new opportunity to achieve coordination. Central to these advances is the idea of self-organization. In the following we show how the paradigm of self-organization can be complemented with the use of ontologies to address design- and communication autonomy in distributed information systems.

Self-Organization. The idea of self-organization is based on the principle that global structures in a complex system can emerge from only local interactions, information and decisions. A self-organizing system functions through contextual local interactions, without any central control. Components aim to individually achieve simple tasks, but a complex collective behavior emerges from their mutual interactions. The system modifies its structure and functionality to adapt to requirements and to the environment based on previous experience. Despite the absence of central control, global information or an external influence, a self-organizing system evolves towards displaying global system behaviors and structures that are more than an aggregation of the properties of its component parts. These global structures are not predefined in advance, instead they are emergent phenomenons. Self-organizing systems typically enjoy failure resilience and scalability, as the coordination is completely decentralized. Nature provides many examples for self organizing processes, including the creation of structures by social animals, such as social insects, or flocking behavior, such as the formation of flocks by birds [CFS+01].

The interest for self organization as a foundation for distributed information systems originates from the fact that these systems need to cope with requirements and constraints stemming from the increased dynamics, autonomy and decentralization – the same or similar characteristics that can be observed in natural systems exhibiting self organization.

Design Autonomy – Emergent Semantics and Meaning Coordination. A key challenge in the development of open distributed information systems is the coordination of meaningful information exchange across nodes that exhibit design autonomy, i.e. autonomously define their schemas to organize their local data. Existing approaches of ontology-based information access almost always assume a setting where information providers share an ontology that is used to access the information. In a decentralized setting with autonomous nodes, this assumption does no longer hold. We rather face the situation where individual nodes maintain their own view of the domain in terms of the organization of the local file system and other information sources. Enforcing the use of a global ontology in such a setting would mean to give up the benefits of the decentralized approach. Recently, a variety of methods have been developed to obtain semantic interoperability in a bottom-up manner, without pre-imposing global models for meaning coordination. In [ACCM+04], the term *emergent semantics* has been used as an umbrella for decentralized approaches to semantic interoperability exploiting the effects of self-organization. The core idea of emergent semantics is that coordination and agreement are emergent phenomenons that arise from interactions. According to [ACCM+04], emergent semantic systems can be characterized by five principles:

1. *Agreements as a semantic handshake protocol.* Meaningful exchanges occur on the basis of mutually accepted propositions. The set of mutual beliefs constitutes the agreements between interacting agents. It is the semantic handshake upon which shared, emerging and dynamic ontologies can be established.

2. *Dynamic agreements emerge from negotiations.* Information exchange is necessary to negotiate new agreements or to verify existing ones. Interaction is required to resolve semantic conflicts, to negotiate and establish consensus on the data interpretation and to verify whether a consensus leads to the expected result.

3. *Agreements emerge from local interactions.* The complexity of emergent semantics and communication costs preclude the option for a node to seek agreements simultaneously with a large number of other nodes; instead communication is kept local. Global agreements are obtained through aggregations of local agreements. As a result, even if a node is only aware of a small fractions of nodes in the system, it will nevertheless be able to interoperate over the whole network indirectly by exploiting aggregate information.

4. *Agreements are dynamic and self-referential approximations.* Since agreements rely on the context of interaction, their boundaries are also fuzzy. Two interacting nodes may achieve an agreement in one application and fail in another, even if the semantic conflicts are the same. Interpretations depend on the context. In turn, agreements are dynamic. Local consensus will be influenced by the context of existing global agreements, thus the process of establishing agreements is self-referential.

5. *Agreements induce semantic self-organization.* In an emergent semantic system, the state space consists of all local communication states reached in consensus building. The attractor, which embodies the global semantic agreement, is obtained when agents locally reach acceptable agreements that are as consistent as possible with the information they receive.

A well-known example in the category of emergent semantic systems is "The Chatty Web" [ACMH03]. The problem of establishing semantic interoperability is viewed as a self-organizing process in which agreements on the interpretation of data are established in a localized, and incremental manner: Nodes provide translations between schemas that are relevant for them and can learn about other translations by routing queries, which is called gossiping. Nodes assess the quality of translations based on three syntactic and semantic criteria: (1) a syntactic analysis of queries after mappings have been applied in order to determine the potential information loss incurred through the transformation, (2) a semantic analysis of composite mappings along cycles in the mapping graph in order to determine the level of agreement that nodes achieve throughout the cycle, and (3) a semantic analysis of search results obtained through composite mappings based on the preservation of data dependencies. The assessment process is incremental and the quality ratings are adjusted along with the operation of the system. In this way, semantic interoperability is increased and semantic agreement at a global level is achieved.

Communication Autonomy – Semantic Network Organization. Due to the communication autonomy of nodes, network organization is a critical problem for the coordination of directing requests to the relevant nodes in a distributed information system. This problem has received particular attention in recent years especially in the Peer-to-Peer community, where a multitude of approaches to network organization have been developed. Generally speaking, all of these approaches rely on building overlay networks on top of an existing network infrastructure to organize and coordinate the communication between nodes, but they vary significantly in their degree of centralization. For example, Distributed Hash Tables (DHTs) allow to efficiently locate data items in large networks based on key-value lookups. Unstructured approaches rely on constrained or unconstrained broadcasting mechanisms to route requests to nodes in the overlay network.

In ontology-based distributed information systems, we have additional knowledge that can be used to further improve the performance of the network organization: Based on the ontology, *semantic overlay networks* can be established, in which links between nodes can be created based on semantic relationships. Such overlays allow to place data nodes semantically together. In general, the idea of placing data nodes close to where relevant queries originate from was already used in early distributed databases [Kos00]. However, these early approaches were based on the assumption that there is a small number of stable nodes, and that there is central control over the organization. Build-

ing semantic overlay networks on top of highly scalable networks without centralized control, opens completely new opportunities.

Edutella [NWS+03] realizes an ontology-based network organization by building concept clusters into a hypercube network topology: Nodes with identical or similar interest are grouped in concept clusters, i.e. specific logical combinations of ontology concepts. Super nodes establish and maintain specific concept clusters. Based on such a topology, queries can be efficiently routed to relevant nodes in the network.

In Chapter 11 we present a more detailed overview of approaches to network organization and develop a new approach for a semantic overlay network that operates in a completely decentralized manner.

4.3 Semantics in Federated Databases

In this and the following sections we show how semantic technologies are applied in the specific classes of distributed information systems and how they address the particular characteristics we identified.

The use of semantics and metadata has a long history in federated database systems. The focus has traditionally been on addressing the heterogeneity of databases. Depending on the respective architecture and the corresponding degree of autonomy, we find three main alternatives to mediate metadata and ontologies across database boundaries [AM99]: In the global approach, component databases agree on a common, federation wide ontology beforehand, and any information sharing and exchange takes place via this global ontology. An approach that allows for more autonomy is based on the use of semantic dictionaries or ontologies, where components agree on a pool of real-world concepts and relationships between concepts. Each component is responsible for expressing the sharable portion of its conceptual schema in terms of the common vocabulary.

As with an increasing number of databases in a federation establishing the correspondences between the sources manually becomes an expensive task, there has been a considerable amount of work on automated schema matching exploiting semantics (c.f. [DLD+04]). The situation becomes even more challenging, if the data sources to be federated do not carry a fixed schema at all. Recent developments such as the InfoSphere project [BHM+05] address this problem by exploiting domain ontologies describing definitions of data classes to be able to classify the data provided by a data source based solely on the properties of their data values.

4.4 Semantics-based Peer-to-Peer Systems

The first successful Peer-to-Peer information systems were systems whose primary focus was that of very simple resource sharing. These resources typically were files, and search for resources was restricted to simple keywords or file names. Efficient algorithms, for example based on DHTs, have been developed to efficiently support key-value lookups. However, it soon turned out that resource descriptions based on keywords and simple key-value lookups were not appropriate to support data sharing beyond simple file sharing.

On the other hand, the Peer-to-Peer paradigm was taken up in the database community, where Peer-to-Peer databases were seen as the natural evolution of federated databases with an increased number of nodes, higher autonomy and higher degree of decentralization. Naturally, the advantages of Peer-to-Peer came with significant challenges for the scalability and interoperability, as many methods developed for federated databases assumed a significant amount of manual design and control.

In Peer-to-Peer systems we are thus again faced with a trade-off between scalability and complexity of the data source descriptions and query languages. Much of the current research thus focuses on pushing the boundaries of this trade-off. It has turned out that semantic technologies are a promising candidate for achieving this goal (c.f. [EHvH+03], [NWQ+02]). In semantics-based Peer-to-Peer systems the shared data itself is represented using ontologies to allow more expressive queries than simple-keyword based queries. The use of semantic descriptions of data sources stored by peers and indeed of semantic descriptions of peers themselves helps in formulating queries such that they can be understood by other peers. The ontologies provide the shared understanding for the communication. Often, these ontologies are not globally imposed, but can be autonomously defined by the individual peers. Mapping formalisms specifically developed for Peer-to-Peer integration allow to express mappings without the use of a global ontology. With the use of emergent semantics it is even possible to deal with changing interpretations. Further, ontologies help to address heterogeneity on the data level, i.e. the problem of dealing with duplicate information, which is a very common problem in Peer-to-Peer information systems.

Besides enabling interoperability, the second place where Peer-to-Peer systems can benefit from semantics is in the network organization: Semantic representation of peer profiles allow to build semantic overlay networks, where the neighborhood mirrors semantic relationships between the peers. In these overlay networks, queries can be efficiently routed to relevant peers.

4.5 Semantic Grid

The Semantic Grid can be seen as a natural evolution of the current Grid towards a knowledge-centric and metadata-driven computing paradigm. The prime driver behind the Semantic Grid effort is the use of explicit semantics for application and information integration as well as describing Grid resources to enable resource discovery tasks ([RBJS03], [DRJS03]). The ability to make meaningful queries over disparate data sources, and to make use of the data in ways which may or may not have been anticipated, requires interoperability of information. For example, this may involve mapping between terminologies used in different domains [RJS05]. This is the classical role of semantic technologies. In the Semantic Grid, ontologies are used for the elaboration, enrichment and annotation of Grid resources, which include users' tasks and needs, data sources, as well as computational resources.

The advance of the Semantic Grid is accelerated by the fact that ontologies, the backbone of semantic technologies, already exist in many application domains, such as bioinformatics. Many semantic technologies can thus be directly applied to the Grid. Yet to achieve full interoperability, semantic technologies need to be applied inside the Grid middleware [RH04].

In e-Science, which is the main application area of the Semantic Grid, much focus is on knowledge discovery and related processes that produce knowledge. Many of these processes involve operations that go beyond the management of information, but instead constitute complex workflows. As a result, next generation Grid architectures introduce a new layer for the management of knowledge to reflect the shift in the focus from managing information to managing knowledge [RBJS03]. The *Knowledge Grid* is concerned with the way that knowledge is acquired, used, retrieved, published and maintained. Related to this shift is the tendency towards service-oriented architectures, where such knowledge producing processes can be managed as first-class objects. Also here semantic technologies provide useful solutions: Semantic Web Services including techniques for service discovery and composition are directly applicable to the next generation Semantic Grid [TDK03].

4.6 Conclusions

In this chapter we have discussed the role of ontologies and semantic technologies in distributed information systems. We have introduced ontologies as explicit specifications of conceptualizations that provide a common and shared understanding of a domain. In distributed information systems, ontologies can be used to formally specify the relationship between the data and its meaning, allowing for a clear and unambiguous interpretation of the data distributed across multiple nodes. We have further analyzed how the use of

ontologies addresses the challenges arising from the characteristics heterogeneity, dynamics and autonomy of nodes:

- *Ontology-based information integration* allows to realize integrated access to heterogeneous data sources.
- *Ontology evolution* allows to manage changes in distributed information systems in a consistent manner.
- *Ontology-based coordination* allows to organize the interactions of autonomous nodes in a decentralized and to an increasing extent self-organizing manner.

Finally, we have shown how the use of these semantic technologies is relevant in federated databases, Peer-to-Peer systems and the Grid.

While in this chapter we have provided a compact overview of the field of semantic technologies in distributed information systems, in the course of this book we will develop and present particular novel methods in detail. In the next chapter we will introduce a specific application scenario of distributed information management, which will serve as a further motivation and running example throughout the work.

Chapter 5

Scenario: Bibster - Sharing Bibliographic Metadata

In this chapter we present one concrete application scenario to motivate and illustrate the use of semantic technologies in distributed information systems. The application scenario addresses the daily life of a researcher, who regularly has to search for publications or their correct bibliographic metadata. The scenario that we envision here is that researchers in a community share bibliographic metadata via a Peer-to-Peer system. The data may have been obtained from BibTeX files or from a bibliography server such as the DBLP database[1]. As one may easily recognize, this scenario exhibits the characteristics that strongly require the use of a decentralized information system: A centralized solution does not exist and cannot exist, because of the multitude of informal workshops that researchers refer to, but that do not show up in centralized resources such as DBLP. Any such centralized resource will only cover a limited scientific community. For example, DBLP covers a lot of Artificial Intelligence, but almost no Knowledge Management, whereas a lot of work is being done in the overlap of these two fields. At the same time, many individual researchers are willing to share their resources, provided they do not have to invest work in doing so.

Furthermore, the scenario exhibits the characteristics we identified for distributed information systems: We are faced with heterogeneity of different forms; while a small common-core ontology of bibliographic information exists (title, author/editor, etc), much of this information is very volatile and users define arbitrary add-ons, for example to include URLs of publications. Because of the semi-structured nature of bibliographic data, there may be many different representations of the same bibliographic entries, detecting duplicates thus is a problem.

The scenario is dynamic in the sense that both the domain (e.g. that of Computer Science literature) as well as the interests of the users in that domain change over time. New

[1] `http://dblp.uni-trier.de/`

concepts (or topics) may become important, others may disappear, and these changes need to be incorporated accordingly.

Finally, in our scenario we find a strong degree of autonomy: While the researchers are typically willing to share their data, they only do so as long as they are able to maintain local control over that data and the sharing does not interfere with their local operations.

This application scenario has been realized and implemented in the Bibster system [HBE+04]. Based on this system, we have performed several evaluation studies to show the benefit of the use of semantic technologies.

This chapter is structured as follows. In Section 5.1 we present major use cases for the Bibster system, in Section 5.2 we illustrate the design of Bibster as a semantics-based Peer-to-Peer system. We demonstrate the use of ontologies and particular semantic technologies in Section 5.3.

5.1 Major Use Cases for Bibster

Bibster is aimed at researchers that share bibliographic metadata. Requirements for Bibster includes capabilities that support their daily work. Researchers may want to

1. query a single specific peer (e.g. their own computer, because it is sometimes hard to find the right entry there), a specific set of peers (e.g. all colleagues at an institute) or the entire network of peers.

2. search for bibliographic entries using simple keyword searches, but also more advanced, semantic searches, e.g. for publications of a special type, with specific attribute values, or about a certain topic.

3. integrate results of a query into a local repository for future use. Such data may in turn be used to answer queries by other peers. They may also be interested in updating items that are already locally stored with additional information about these items obtained from other peers.

The screenshot in Figure 5.1 partially indicates how these use cases are realized in Bibster. The *Scope* widget allows for defining the targeted peers, the *Search* and *Search Details* widgets allow for keyword and semantic search; *Results Table* and *BibtexView* widgets allow for browsing and re-using query results. The query results are visualized in a list grouped by duplicates. They may be integrated into the local repository or exported into formats such as BibTeX and HTML.

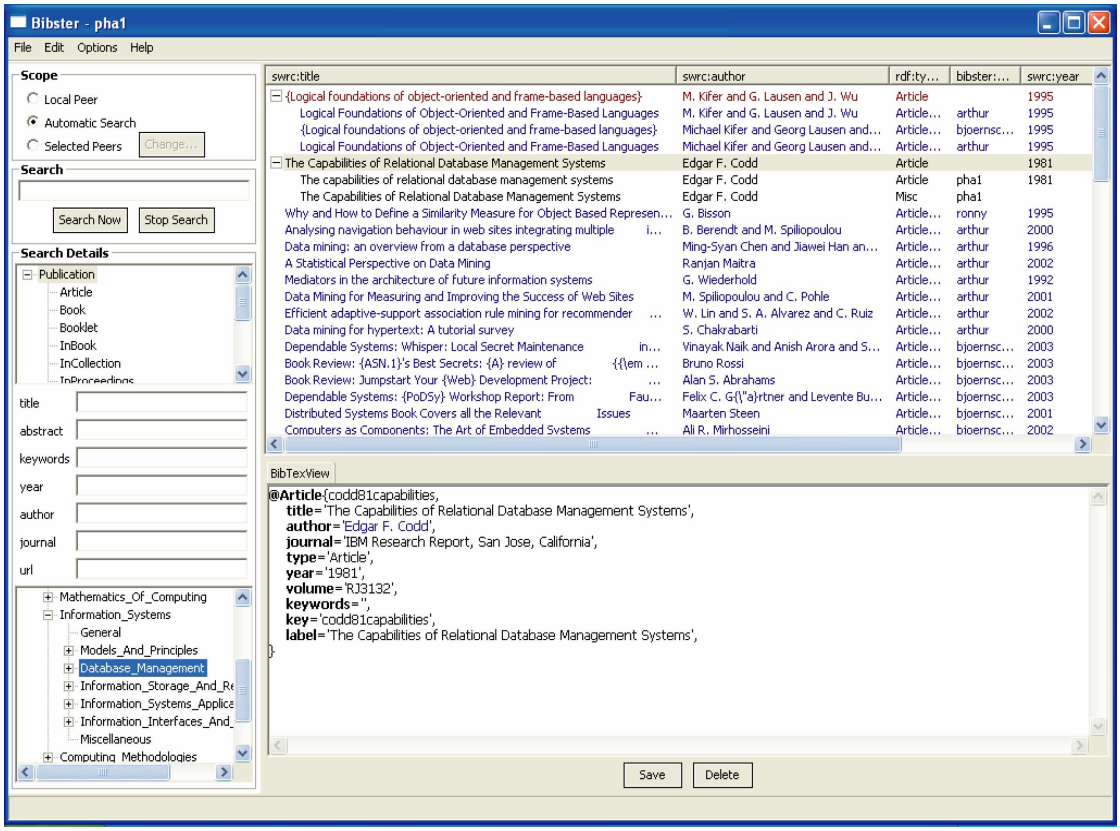


Figure 5.1: Bibster Screenshot

5.2 Design of Bibster

The Bibster system has been implemented as an instance of the Swapster System architecture as introduced in [EHvH+03]. Swapster was developed in the SWAP project as a generic platform to account for the general need of sharing semantics-based information in a Peer-to-Peer fashion. Figure 5.2 shows a high-level design of the architecture of a single node in the Peer-to-Peer system.

Communication Adapter: This component 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. In the specific implementation of the Bibster system we use JXTA [TAD+02] as the communication platform.

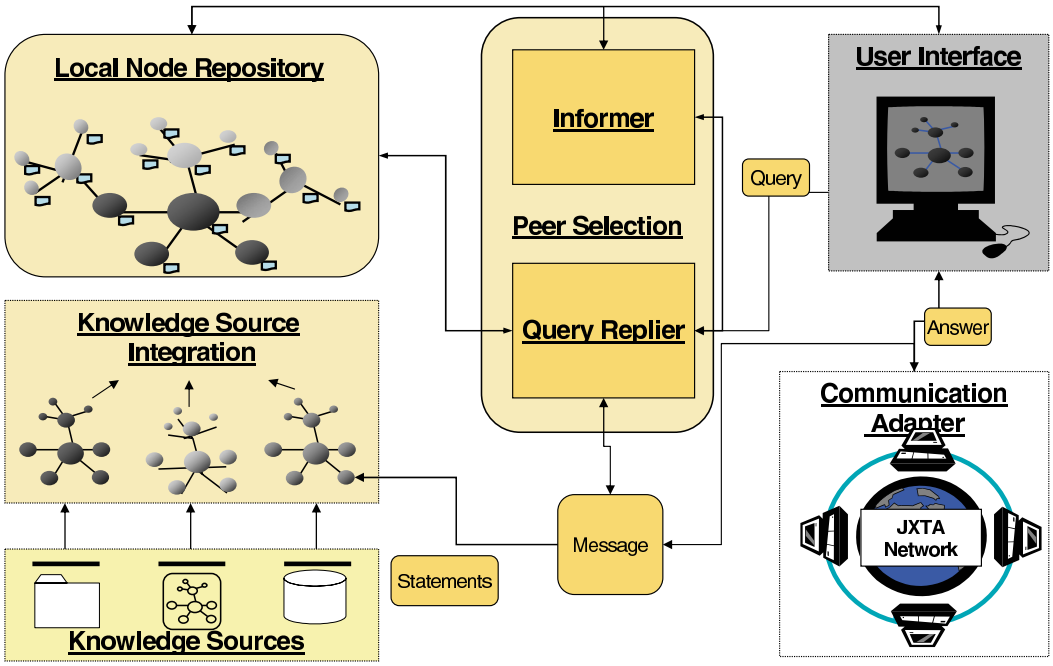


Figure 5.2: SWAP System Architecture

Knowledge Sources: The knowledge sources in the Bibster system are sources of bibliographic metadata, such as BibTeX files stored locally in the file system of the user. In our terminology of distributed information systems, the knowledge sources in the SWAP architecture actually correspond to data sources. The term knowledge sources has been chosen due to the focus of SWAP on knowledge management applications. In this sense, the data sources in fact are the original "source of the knowledge".

Knowledge Source Integrator: The Knowledge Source Integrator is responsible for the extraction and integration of internal and external knowledge sources into the Local Node Repository. In Section 5.3.1 we describe the process of ontology-based integration of local and remote sources in Bibster and how different aspects of heterogeneity are addressed, for example how duplicate query results are detected.

Local Node Repository: In order to manage its local information as well as information acquired from remote sources, each peer maintains a Local Node Repository

providing the following functionality: (1) Mediate between different information models, (2) support query formulation and processing, (3) specify the peer's interface to the network, and (4) provide the basis for peer selection. In the Bibster system, the Local Node Repository is based on the Sesame Repository [BKvH01].

Informer: The task of the Informer is to pro-actively advertise the available information of a peer in the Peer-to-Peer network and to discover peers with information that may be relevant for answering the user's queries. This is realized by sending advertisements about the expertise of a peer. In the Bibster system, these expertise descriptions contain a set of topics that the peer is an expert on. Peers may accept – i.e. remember – these advertisements, thus creating a semantic link to the other peer. These semantic links form a semantic overlay network, which is the basis for intelligent query routing.

Query Replier: The Query Replier is the coordinating component controlling the process of distributing queries. It receives queries from the User Interface or from other peers. In both cases it tries to answer the query or distribute it further according to the content of the query. The decision to which peers a query should be sent is based on the knowledge about the expertise of other peers.

User Interface: The User Interface (Figure 5.1) allows the user to import, create and edit bibliographic metadata as well as to easily formulate queries.

5.3 Ontologies in Bibster

Ontologies are crucial throughout the usage of Bibster, namely for importing data, formulating queries, routing queries, and processing answers.

First, the system enables users to import their own bibliographic metadata into a local repository. Bibliographic entries made available to Bibster by a user are automatically aligned to two ontologies: The first ontology (SWRC [SBH+05]) describes different generic aspects of bibliographic metadata (and would be valid across many different research domains), the second ontology (ACM Topic Ontology[2]) describes specific categories of literature for the Computer Science domain.

Second, queries are formulated in terms of the two ontologies: Queries may concern fields like author, publication type, etc. (using terms from the SWRC ontology) or queries may concern specific Computer Science terms (using the ACM Topic Ontology).

Third, queries are routed through the network depending on the expertise models of the peers describing which concepts from the ACM ontology a peer can answer queries

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

on. A matching function determines how closely the semantic content of a query matches the expertise model of a peer. Routing is then done on the basis of this semantic ranking.

Last, answers are returned for a query. Due to the distributed nature and potentially large size of the Peer-to-Peer network, this answer set might be very large and contain many duplicate answers. Because of the semi-structured nature of bibliographic metadata, such duplicates are often not exactly identical copies. Ontologies help to measure the semantic similarity between the different answers and to remove apparent duplicates as identified by the similarity function.

In the following, we will present the use of ontologies in Bibster and corresponding semantic technologies along the dimensions of ontology-based information integration, ontology evolution and ontology-based coordination.

5.3.1 Integrating Bibliographic Metadata

Representation of Bibliographic Metadata with the SWRC Ontology Many researchers have accumulated extensive collections of BibTeX files for their bibliographic references. However, these files are semi-structured and thus single attributes may be missing or may not be interpreted correctly.

To enable interoperability between nodes in the Peer-to-Peer system, the bibliographic metadata is represented using an ontology. We here follow a transformation-based approach: Plain BibTeX files are lifted into an ontology-based representation with *BibToOnto*[3]. The target ontology is the Semantic Web Research Community Ontology (SWRC), which models among others a research community, its researchers, topics, publications, tools, and properties between them. The SWRC ontology defines a shared and common domain theory, which helps users and machines to communicate concisely, and supports the exchange of semantics-based metadata.

The SWRC ontology generically models key entities relevant for typical research communities and the relations between them. The current version of the ontology comprises a total of 53 concepts in a taxonomy and 42 object properties, 20 of which are participating in 10 pairs of inverse object properties. All entities are enriched with additional annotation information.

The SWRC ontology comprises at total of six top level concepts, namely the `Person`, `Publication`, `Event`, `Organization`, `Topic` and `Project` concepts. Figure 5.3 shows a small portion of the SWRC ontology with its main top-level concepts and relations[4]. The `Person` concept models any kind of human person, and a large number of properties restrict their domain or range to individuals of this concept like `worksAtProject` or `author`, respectively. The `Person`

[3] `http://bibtoonto.sourceforge.net/`

[4] The ontology is visualized according to the UML Profile of the OWL Ontology Definition Metamodel originally presented in [BVEL04] and extended in [BHHS06].

concept is specialized by a large number of – not necessarily disjoint – subconcepts, e.g. `Employee`, `Student` and the like. The `Event` concept is meant to model different types of events and is thus specialized by a wide range of concepts including events like `Lecture` or `Conference`. The `Publication` concept subsumes all different types of research publications modeled in close correspondence with the well known BibTeX publication types like `Article` or `Proceedings`. The `Organization` and `Project` concepts model more abstract concepts like the subconcepts `Department` or `SoftwareProject`, respectively. The `Topic` captures arbitrary topics of interest, as explained in the following.

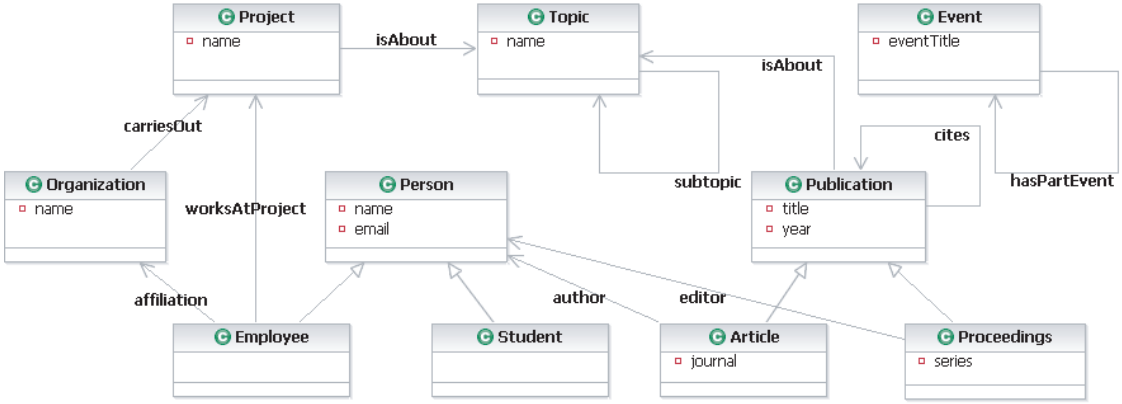


Figure 5.3: Main Concepts of the SWRC Ontology

Representation of Topic Classification. A key feature of bibliographic metadata is the classification against established topic hierarchies. In a sense, topic hierarchies can be seen as light-weight ontologies by themselves that complement the SWRC ontology with domain specific models of particular research domains. One example is the ACM topic ontology for the Computer Science domain. Further, many other topic hierarchies for other domains exist, for which an integration with the SWRC ontology is desirable.

In alignment with the choice of our ontology language, which requires a strict separation between concepts and instances, we model topics as instances of the concept `Topic`. The instances of `Topic` are arranged hierarchically by means of the `subtopic` relation. The relation between publications and a topic is established via the `isAbout` property. To link a specific topic ontology with the SWRC ontology, the topic concept of the topic ontology, e.g. for the ACM topic ontology: `ACMTopic`, specializes the `Topic` concept of the SWRC ontology `Topic`.

BibToOnto automatically classifies bibliographic entries according to the ACM topic ontology using a simple keyword based approach [HSvH04]. Additionally, it is possible to reclassify the entries manually in the user interface of Bibster. The ACM topic ontology is a standard schema for describing and categorizing computer science litera-

ture. It covers 1287 topics of the computer science domain. In addition to the sub- and supertopic relations, it also provides information about related topics.

The following example shows a transformation of a BibTeX entry to an SWRC ontology-based item. The result[5] is shown in Figure 5.4.

Example 5
```
@ARTICLE{codd70relational
        author    = "Edgar F. Codd",
        year      = "1970",
        title     = "A relational model for
                      large shared data banks",
        journal   = "Communications of ACM",
        volume    = "13",
        number    = "6",
        pages     = "377--387"}
```

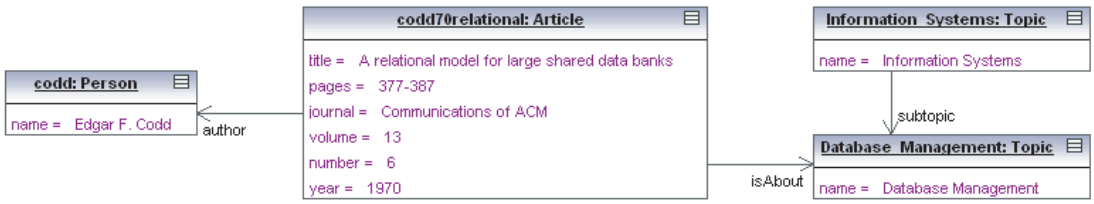


Figure 5.4: SWRC Sample Metadata

Semantic Querying Each peer node in the Bibster system manages a local repository with bibliographic data extracted by BibToOnto or integrated from other peers. This repository provides an interface for query answering against the data with the semantics provided by the ontology.

In our running example, a researcher is querying for publications written by the author Codd about database management. Let us look at the corresponding query in SPARQL[6]:

Example 6
```
PREFIX swrc:  <http://swrc.ontoware.org/ontology#>
PREFIX acm:<http://daml.umbc.edu/ontologies/topic-ont#>
```

[5]For better readability we used a concatenation of the author name and the title of the publication as identifiers in this example. In the Bibster system however we calculate hash codes over all attribute values to guarantee the uniqueness of URIs.

[6]In the implementation of the original Bibster system, we used SeRQL [Bro05] – a predecessor of SPARQL – as a query language. For consistency of this work, we provide examples in SPARQL. For the matter of the queries under consideration, this is a purely syntactic difference.

```
PREFIX topic: <http://daml.umbc.edu/ontologies/classification#>

CONSTRUCT {?pub swrc:title ?title; swrc:author ?author }
FROM { ?pub rdf:type swrc:Publication;
  swrc:title ?title;
  acm:topic <topic:ACMTopic/Information_Systems/Database_Management>
  swrc:author ?author . ?author swrc:lastName "Codd"
}
```

This query demonstrates several characteristics of semantic query languages that are required in the context of Bibster. We here briefly outline the importance of some of the properties of ontology query languages as analyzed in Chapter 3.3:

1. *The combination of different ontologies:* The user is able to formulate queries against multiple ontologies in this case the combination of SWRC and ACM topic ontology.

2. *The use of the semantics defined by the ontology:* The query takes the semantics of the underlying ontologies into account. As a very simple example, the ontology defines the concept `Article` to be a subconcept of `Publication`, the query will therefore return the article of Example 5.

3. *The closure of the query language:* Closure here refers to the fact that the query operates on an ontology and returns an ontology as a result (as defined by the `CONSTRUCT` pattern), which may be integrated into the local repository, sent to other peers, or queried again.

Semantic Duplicate Detection When querying the Bibster network one receives a large number of results with an often high number of duplicates. This is due to the fact that no centralized but many distributed local repositories have been used. Furthermore, the representation of the metadata is very heterogeneous and possibly even contradicting. To enable an efficient and easily usable system, Bibster presents query results grouping duplicates together. Duplicates in Bibster are bibliographic entries which refer to the same publication, person, or organization in the real world, but are modeled as different resources. Bibster uses specific similarity functions to recognize two resources as being duplicates. For the calculation of the similarity we apply the framework for similarity in ontologies, which will be presented in Chapter 7.

For each resource type (publication, person, organization), a set of specific features used to assess the similarity between two of its instances has been compiled. For instance, publications are assessed based on their titles, publication types, authors, years, ACM topics, etc. For each of the features we use different *individual similarity functions*, which are grouped as follows:

Individual similarity functions have been used on a *data value level*, an *ontology level*, and *background knowledge about the bibliographic domain* as described in detail in Chapter 7. From the variety of individual similarity functions, an overall value is obtained with an aggregated similarity function, using a weighted average over the individual functions.

For Bibster, the weights have been assigned based on experiments with sample data. More precisely, several duplicates were detected manually. From these training duplicates the weights were adjusted to achieve a maximal f-measure (combination of precision and recall) value.

Those pairs of resources are considered as duplicates, whose similarity is larger than a certain threshold. Instead of presenting all individual resources of the query result, duplicates are visualized as one, merged, resource. The merged resources comprise the union of statements of the individuals identified as duplicates. In the case of conflicting property values, heuristics for the merging of resources are applied (e.g. for book titles to select the most detailed value with the least abbreviations).

5.3.2 Collaborative Ontology Evolution

In the initial Bibster system we assumed a globally shared and static ontology. While such an assumption is practical for the domain ontology part modeled by the SWRC ontology, it poses a serious restriction for the topic ontology that is used to classify the bibliographic metadata. Already the sheer size of the ACM topic ontology makes it quite difficult for users to easily locate topics which are relevant for them. We therefore extended the Bibster system to alleviate this situation and allow for personal ontologies that can be adapted and changed over time by the user.

Evolution of Personal Ontologies These personal ontologies reflect the interests of users at certain times. Interests might change as well as the available data, therefore the personalization requires quite naturally support for the evolution of personal ontologies. For example, a particular user might become interested in the topic of *Heterogeneous Databases* and decide to add it to his personal ontology as a subtopic of *Database Management*. Such changes obviously require additional coordination in the system, but in turn, we can also benefit from the changes performed by the community of users by recommending relevant ontology elements to other users with similar interests. For example, the topic of *Heterogeneous Databases* might potentially be relevant to users who have previously asked queries about similar topics.

Collaborative Filtering for Ontology Evolution Of particular interest are therefore collaborative filtering systems which can produce personal recommendations by computing the similarity between own preferences and the one of other peers. In Bibster,

we therefore adapted a collaborative filtering recommender system to assist users in the management and evolution of their personal ontology by providing detailed suggestions of ontology changes. The approach has been thoroughly evaluated with very promising results, as we show in detail in Chapter 9.

5.3.3 Coordination with Expertise-Based Peer Selection

The scalability of a Peer-to-Peer network is essentially determined by the way how queries are propagated in the network. Peer-to-Peer networks that broadcast all queries to all peers do not scale – intelligent query routing and network topologies are required to be able to route queries to a relevant subset of peers that are able to answer the queries. Here we give an overview of the model of expertise based peer selection and how it is used in the Bibster system. A detailed description can be found in Chapter 11.

In this model, peers use an ontology to advertise semantic descriptions of their expertise in the Peer-to-Peer network. The knowledge about the expertise of other peers forms a semantic overlay network, independent of the underlying network topology. If the peer receives a query, it can decide to forward it to peers about which it knows that their expertise is similar to the subject of the query.

Semantic Description of Expertise. The Peer-to-Peer network consists of a set of peers. Every peer has a Local Node Repository, which stores the bibliographic metadata. The peers use a topic ontology to describe the expertise of peers and the subject of queries. An expertise description is an abstract, semantic description of the Local Node Repository of a peer based on this ontology. The expertise of a peer is thus a set of topics, for which a peer provides classified instances.

Advertisements are used to promote descriptions of the expertise of peers in the network. An advertisement associates a peer with its expertise. Peers decide autonomously – without central control – whom to promote advertisements to and which advertisements to accept. This decision is based on the semantic similarity between expertise descriptions.

Matching and Peer Selection. Queries are posed by a user and are evaluated against the local node repositories of the peers. First a peer evaluates the query against its local node repository and then decides which peers the query should be forwarded to. A subject is an abstraction of a given query expressed in terms of the common ontology. The subject specifies the required expertise to answer the query. In our scenario, the subjects of queries are the set of topics that are referenced in the query. For instance, the extracted subject of the query in Example 6 would be *Information Systems/Database Management.*

Again, a similarity function yields the semantic similarity between a subject and an expertise description. The similarity is used for determining to which peers a query should be forwarded. In Bibster, the similarity function is based on the idea that topics which are close according to their positions in the topic ontology are more similar than topics that have a larger distance. For example, an expert on the topic *Information Systems/Information Storage and Retrieval* has a higher chance of giving a correct answer on a query about *Information Systems/Database Management* than an expert on a less similar topic like *Hardware/Memory Structures*.

The peer selection algorithm returns a ranked set of peers, where the rank value is equal to the similarity value provided by the similarity function. Therefore, peers that have an expertise more similar to that of the subject of the query will have a higher rank.

Semantic Overlay Network. The knowledge of the peers about the expertise of other peers is the basis for a semantic overlay network that is built on top of the initially unstructured underlying network. The acquaintances between peers are established by the selection of which peers a peer sends its advertisements to. Furthermore peers can decide to accept an advertisement, e.g. to include it in their registries, or to discard the advertisement. The semantic overlay network in combination with the expertise based peer selection is the basis for intelligent query routing.

5.4 Conclusions

In this chapter, we have introduced the application scenario of exchanging bibliographic metadata to motivate and illustrate the use of semantic technologies in distributed information systems. We further described the design and implementation of Bibster, a semantics-based Peer-to-Peer that implements this application scenario. Bibster exploits lightweight ontologies in all its crucial aspects: data-organization, query formulation, query routing, and duplicate detection. Bibster constitutes the first ontology-based Peer-to-Peer systems deployed in the field.

The Bibster system has been evaluated by means of a public field experiment (June to August 2004). The user actions and system events were continuously logged and analyzed to evaluate the user behavior and system performance. We will report on various aspects of the evaluation results in the respective subsequent chapters.

Summarizing some experiences gained from the development and application of Bibster, it is obvious that for Bibster and similar applications the usage of semantic technologies and ontologies provide an *added value*. Semantic structures serve important user concerns like high quality duplicate detection or comprehensive searching capabilities.Unsurprisingly, in small networks with *small user groups*, intelligent query routing is not a major issue. While it is beneficial to direct queries to specific peers known to

the user, advanced routing algorithms may only be beneficial for a much larger number of users in a network.

Based on the SWAP system architecture, several other instantiations have been built. These include Oyster [PH05], a Peer-to-Peer system for sharing metadata about ontologies with the goal to foster ontology re-use in communities and Xarop [TEF+04], a knowledge sharing system for virtual organizations.

Part II

Ontology-based Information Integration

Chapter 6

A Mapping System for Distributed Ontologies

To enable interoperability between nodes in large distributed information systems based on heterogeneous ontologies, it is necessary to specify how the data residing at a particular node corresponds to data residing at another node. This is formally done using the notion of a mapping. There are three lines of work connected to the problem of mapping: (1) identifying correspondences between heterogeneous data sources, (2) representing these correspondences in an appropriate mapping formalism, and (3) using the mappings for a given integration task.

In this chapter, we focus on the latter two important problems once the correspondences between data sources are known: those of representing the mappings using an appropriate formalism and using them for some specific integration tasks, such as data transformation or query answering over heterogeneous data sources. An immediate question is whether the OWL language itself provides the solutions to these problems, as OWL already allows to express simple mappings between ontology elements. We argue that the OWL ontology language itself is not sufficient for expressing mappings in distributed information systems. In many scenarios, different ontologies are built by different individuals for different purposes. We must thus expect the same information to be represented in different forms and with different levels of abstraction in the various ontologies. When mapping elements in the various ontologies to each other, it may happen that an element in one ontology corresponds to a view (i.e., a query) over the other ontologies. In our work, we follow the general framework of [Len02] to formalize the notion of a mapping system for OWL DL ontologies, where mappings are expressed as correspondences between queries over ontologies. It is easy to see that query answering within such a mapping system is undecidable. To obtain an alternative more suitable for practical applications, we introduce restrictions required to attain decidability. These restricted, but still very expressive mappings, can be expressed in OWL DL extended with the so-called *DL-safe* subset of the Semantic Web Rule Language (SWRL) [MSS04].

Furthermore, we note that these restrictions can be relaxed for so-called *tree-like mappings* using query roll-up techniques from [HT02].

In Section 6.1 we present an overview of existing approaches for representing mappings, their application for particular integration tasks and their limitations with respect to ontology-based information integration. Based on the ontology model formally introduced in Section 6.2, we present the formalization of our mapping system in Section 6.3. We demonstrate the expressiveness of the mapping system with a practical example from our bibliographic domain. While possible applications of the mapping system are manifold, including for example data transformation and data exchange [FKMP03], we show in Section 6.4 how our proposed mapping system can be applied for the task of *ontology integration*, which addresses the problem of integrating a set of local ontologies. Queries are expressed against a global, integrated ontology, which provides a unified view of the local ontologies. Query answering in the ontology integration system (using conjunctive queries) is based on a novel technique for reducing description logic knowledge bases to disjunctive datalog programs [HMS04, Mot06]. After a discussion of related work in Section 6.5, we summarize and conclude in Section 6.6.

6.1 An Overview of Mapping Formalisms

Mappings provide the foundations for many applications that require some sort of integration of heterogeneous data sources. In this section we analyze different applications of mappings, present a mapping formalism that allows a representation suitable for all of these tasks and finally discuss approaches to one of the applications, that of data integration – in more detail.

6.1.1 Mapping Applications

Data Integration Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of the data [Len02]. The process of creating this view is called schema integration. Typical data integration systems are characterized by an architecture based on a set of sources that contain the data and a global schema that provides a reconciled and integrated view of the underlying sources. The mapping is used to specify the correspondences between the sources and the global schema.

Data Exchange In alignment with [FKMP03], we define data exchange as the problem of taking data structured under a source ontology, and creating an instance of a target ontology that reflects the source as accurately as possible. Similarly to data integration, the relation between source and target schema can be described using mappings. However, there are certain differences: [FKMP03] compares data integration with data exchange

and shows how a data exchange setting can be thought of as a data integration setting in terms of the notation of [Len02]. The major difference pointed out is that in a data exchange setting, a finite target instance that best reflects the given source instance needs to be *materialized*, whereas in data integration no such exchange of data is required. For query answering in data integration, the queries against the global schema are evaluated using the source instances, in data exchange, queries are evaluated using the materialized target instances.

6.1.2 Mapping System

We consider a very general form of mapping that subsumes virtually all existing mapping formalisms used for a variety of applications:

Definition 1 (Mapping System) *A mapping system is a triple $(\mathcal{S}, \mathcal{T}, \mathcal{M})$, where $\mathcal{S}$ and $\mathcal{T}$ are the source and target schema and $\mathcal{M}$ is a set of mapping assertions between $\mathcal{S}$ and $\mathcal{T}$ of the form $q_S \rightsquigarrow q_T$, where q_S and q_T are queries over $\mathcal{S}$ and $\mathcal{T}$.*

At this point, we do not make any further assumptions about the representation of the source and target schema. Classically, they are represented as relational schemas. Further, the queries are usually restricted to queries that return sets of tuples, and the $\rightsquigarrow$ relation is either a subset ($\subseteq$, $\supseteq$) or equals ($=$) relation.

6.1.3 Data Integration using the Mapping System

One of the most important design decisions for data integration systems is the way how the correspondences between the schemas of the source schemas and the global schema – i.e. the target schema – are specified. In general, there are two main approaches to model these relationships: Global-As-View (GAV) requires that the global schema is expressed in terms of the data sources. Local-As-View (LAV) requires the global schema to be specified independently from the sources, while every source is defined as a view over the global schema.

Local-As-View. In a LAV data integration system, the mapping $\mathcal{M}$ associates to each element s of $\mathcal{S}$ a query $q_\mathcal{T}$ over $\mathcal{T}$. That means, the mapping $\mathcal{M}$ contains a set of assertions (one for each s) of the form

$$s \rightsquigarrow q_\mathcal{T}$$

Obviously, the LAV-approach is favorable, if the global schema is stable and well-established (e.g. a shared ontology). The LAV-approach also favors extensibility, as adding a new source simply requires adding the new corresponding assertions, without other changes.

Query processing in LAV systems is difficult, as the answer to a query is to be computed over a set of views (*view based query processing*), instead of over the data itself. Two approaches are *view-based query rewriting* and *view-based query answering*. [Len02], [AD98] present complexity results for query answering under certain assumptions (e.g. CWA vs. OWA) and for certain languages (e.g. conjunctive queries, positive queries, Datalog). [BLR97] presents theoretical results for the problem of query rewriting.

Global-As-View. In a GAV data integration system, the mapping $\mathcal{M}$ associates to each element t of $\mathcal{T}$ a query $q_{\mathcal{S}}$ over $\mathcal{S}$. That means, the mapping $\mathcal{M}$ contains a set of assertions (one for each t) of the form

$$t \rightsquigarrow q_{\mathcal{S}}$$

Obviously, the GAV-approach is favorable, if the set of sources is stable. As the assertions already define how to retrieve the data, query processing in GAV is straight forward. On the other hand, adding new sources to the system may be complicated, as the new source may have impact on the existing definitions.

6.1.4 Limitations of Existing Mapping Formalisms

The tasks of data integration, data translation etc have long been studied in the database community. Many of the findings are directly applicable to ontology-based information integration. However, there are certain differences with classical data integration that need to be considered in the context of ontology-based information systems:

Representation of source and target The main difference between classical data integration in databases and ontology-based information integration lies in the characteristics of the structures that are being mapped: In databases, typically source and target are described as simple relational schemas, in the most simple case a finite collection of relation symbols. In some cases, these relational schemas carry additional constraints, e.g. in the form of dependencies over the schemas. In ontology-based information integration, the structures that are being mapped are themselves ontologies, i.e. sentences in rich Description Logics.

Representation of mappings Most mapping formalisms rely on the notion of a global schema that is used to provide a unified view over a set of sources. In decentralized settings, such a global schema may not exist, instead we require the ability to express mappings in arbitrary directions in a network of ontologies. In the literature, some approaches combining features of GAV and LAV have been proposed. For example, [FLM99] presents the Global-and-Local-As-View approach (GLAV), which allow assertions of the form $q_S \rightsquigarrow q_T$, where q_S and q_T are conjunctive queries over $\mathcal{S}$ and $\mathcal{T}$. An

approach to Peer-to-Peer data integration based on epistemic logic has been described in [KAM03]. However, mappings that go beyond GAV or LAV are not well established so far.

A further issue regarding the representation of mappings is whether they can be treated as "first-class citizens" with a proper representation language that allows mappings to be exchanged just as any other information.

6.2 Formal Ontology Model

In this section we formally define the syntax and semantics of the $\mathcal{SHOIN}(\mathbf{D})$ description logics as well as conjunctive queries and rule extensions over $\mathcal{SHOIN}(\mathbf{D})$ description logics. While we have already informally introduced the concepts in Chapter 3, this formalization is required for the definition of the semantics of the mapping system.

6.2.1 $\mathcal{SHOIN}(\mathbf{D})$ Description Logic

We now formally define the syntax and semantics of the $\mathcal{SHOIN}(\mathbf{D})$ description logic:

Definition 2 *Let N_C be a set of* concept names*, N_{R_a} and N_{R_c} sets of* abstract and concrete role names*, respectively, and N_{I_a} and N_{I_c} sets of* abstract and concrete individuals*, respectively. An* abstract role *is an abstract role name or the inverse S^- of an abstract role name S (concrete roles do not have inverses). Finally, let $\mathbf{D}$ be an admissible concrete domain.*

An RBox *$KB_\mathcal{R}$ is a finite set of transitivity axioms $\mathsf{Trans}(R)$, and role inclusion axioms of the form $R \sqsubseteq S$ and $T \sqsubseteq U$, where R and S are abstract roles, and T and U are concrete roles. The reflexive-transitive closure of the role inclusion relationship is denoted with $\sqsubseteq^*$. A role not having transitive subroles (w.r.t. $\sqsubseteq^*$, for a full definition see [HST00]) is called a* simple *role.*

The set of $\mathcal{SHOIN}(\mathbf{D})$ concepts *is defined by the following syntactic rules, where A is an atomic concept, R is an abstract role, S is an abstract simple role, $T_{(i)}$ are concrete roles, d is a concrete domain predicate, a_i and c_i are abstract and concrete individuals, respectively, and n is a non-negative integer:*

$$\begin{aligned} C &\rightarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \\ &\quad \geq n\, S \mid \leq n\, S \mid \{a_1, \ldots, a_n\} \mid \geq n\, T \mid \leq n\, T \mid \\ &\quad \exists T_1, \ldots, T_n.D \mid \forall T_1, \ldots, T_n.D \\ D &\rightarrow d \mid \{c_1, \ldots, c_n\} \end{aligned}$$

A TBox *$KB_\mathcal{T}$ is a finite set of concept inclusion axioms $C \sqsubseteq D$, for C and D concepts; an* ABox *$KB_\mathcal{A}$ is a finite set of concept and role assertions and individual*

Table 6.1: Translation of $\mathcal{SHOIN}(\mathbf{D})$ into FOL

Mapping Concepts to FOL	
$\pi_y(\top, X)=\top$	$\pi_y(\bot, X)=\bot$
$\pi_y(A, X)=A(X)$	$\pi_y(\neg C, X)=\neg\pi_y(C, X)$
$\pi_y(C \sqcap D, X)=\pi_y(C, X) \wedge \pi_y(D, X)$	$\pi_y(C \sqcup D, X)=\pi_y(C, X) \vee \pi_y(D, X)$
$\pi_y(\forall R.C, X)=\forall y : R(X, y) \rightarrow \pi_x(C, y)$	$\pi_y(\exists R.C, X)=\exists y : R(X, y) \wedge \pi_x(C, y)$
$\pi_y(\{a_1 \ldots, a_n\}, X)=X \approx a_1 \vee \ldots \vee X \approx a_n$	$\pi_y(\{c_1^{\mathsf{c}}, \ldots, c_n^{\mathsf{c}}\}, X)=X \approx_{\mathbf{D}} c_1^{\mathsf{c}} \vee \ldots \vee X \approx_{\mathbf{D}} c_n^{\mathsf{c}}$
$\pi_y(d, X_1, \ldots, X_m)=$	$d(X_1, \ldots, X_m)$
$\pi_y(\leq n\, R.C, X)=$	$\forall y_1, \ldots, y_{n+1} : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \rightarrow \bigvee y_i \approx y_j$
$\pi_y(\geq n\, R.C, X)=$	$\exists y_1, \ldots, y_n : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \wedge \bigwedge y_i \not\approx y_j$
$\pi_y(\forall T_1, \ldots, T_m.D, X)=$	$\forall y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge T_i(X, y_i^{\mathsf{c}}) \rightarrow \pi_x(D, y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\pi_y(\exists T_1, \ldots, T_m.D, X)=$	$\exists y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge T_i(X, y_i^{\mathsf{c}}) \wedge \pi_x(D, y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\pi_y(\leq n\, T, X)=$	$\forall y_1^{\mathsf{c}}, \ldots, y_{n+1}^{\mathsf{c}} : \bigwedge T(X, y_i^{\mathsf{c}}) \rightarrow \bigvee y_i^{\mathsf{c}} \approx_{\mathbf{D}} y_j^{\mathsf{c}}$
$\pi_y(\geq n\, T, X)=$	$\exists y_1^{\mathsf{c}}, \ldots, y_n^{\mathsf{c}} : \bigwedge T(X, y_i^{\mathsf{c}}) \wedge \bigwedge y_i^{\mathsf{c}} \not\approx_{\mathbf{D}} y_j^{\mathsf{c}}$
Mapping Axioms and KB to FOL	
$\pi(C \sqsubseteq D)=$	$\forall x : \pi_y(C, x) \rightarrow \pi_y(D, x)$
$\pi(R \sqsubseteq S)=$	$\forall x, y : R(x, y) \rightarrow S(x, y)$
$\pi(\mathsf{Trans}(R))=$	$\forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
$\pi(C(a))=$	$\pi_y(C, a)$
$\pi(R(a, b))=$	$R(a, b)$
$\pi(a^{(\mathsf{c})} \circ b^{(\mathsf{c})})=$	$a \circ_{(\mathbf{D})} b$ for $\circ \in \{\approx, \not\approx\}$
$\pi(KB)=$	$\bigwedge_{R \in N_R} \forall x, y : R(x, y) \leftrightarrow R^-(y, x) \wedge \bigwedge_{\alpha \in KB_{\mathcal{R}} \cup KB_{\mathcal{T}} \cup KB_{\mathcal{A}}} \pi(\alpha)$

X is a meta variable and is substituted with the actual variable. π_x is obtained from π_y by simultaneously substituting all $y_{(i)}$ with $x_{(i)}$ and π_y with π_x, and vice versa.

(in)equalities $C(a)$, $R(a, b)$, $a \approx b$ and $a \not\approx b$, respectively. A $\mathcal{SHOIN}(\mathbf{D})$ knowledge base *KB is a triple $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, KB_{\mathcal{A}})$.*

The semantics of a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base KB is defined by the mapping π that translates KB into a first-order formula as specified in Table 6.1.

The $\mathcal{SHIQ}(\mathbf{D})$ description logic is obtained from $\mathcal{SHOIN}(\mathbf{D})$ by disallowing nominals and introducing qualified number restrictions.

6.2.2 Rules and Conjunctive Queries

We now introduce the notion of conjunctive queries over a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base KB. This notion is used in Section 6.3 to define the mapping formalism.

Definition 3 (Conjunctive Queries) *Let KB be a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base, and let N_P be a set of predicate symbols, such that all $\mathcal{SHOIN}(\mathbf{D})$ concepts and all abstract and concrete roles are in N_P. An* atom *has the form $P(s_1, \ldots, s_n)$, often denoted as $P(\mathbf{s})$, where $P \in N_P$, and s_i are either variables or individuals from KB. An atom is called a* DL-atom *if P is a $\mathcal{SHOIN}(\mathbf{D})$ concept, or an abstract or a concrete role; it is called* non-DL-atom *otherwise.*

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ be sets of distinguished *and* non-distinguished *variables, denoted as $\mathbf{x}$ and $\mathbf{y}$, respectively. A* conjunctive query *over KB, written as*

$Q(\mathbf{x}, \mathbf{y})$, *is a conjunction of atoms* $\bigwedge P_i(\mathbf{s_i})$, *where the variables in* $\mathbf{s_i}$ *are contained in either* $\mathbf{x}$ *or* $\mathbf{y}$.

A conjunctive query $Q(\mathbf{x}, \mathbf{y})$ *is* DL-safe *if each variable occurring in a DL-atom also occurs in a non-DL-atom in* $Q(\mathbf{x}, \mathbf{y})$.

We extend the operator π *from Section 6.2.1 to translate* $Q(\mathbf{x}, \mathbf{y})$ *into a first-order formula with free variables* $\mathbf{x}$ *as follows:*

$$\pi(Q(\mathbf{x}, \mathbf{y})) = \exists \mathbf{y} : \bigwedge \pi(P_i(\mathbf{s_i}))$$

For $Q_1(\mathbf{x}, \mathbf{y_1})$ *and* $Q_2(\mathbf{x}, \mathbf{y_2})$ *conjunctive queries, a* query containment *axiom* $Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1})$ *has the following semantics:*

$$\begin{aligned} &\pi(Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1})) = \\ &\quad \forall \mathbf{x} : \pi(Q_1(\mathbf{x}, \mathbf{y_1})) \leftarrow \pi(Q_2(\mathbf{x}, \mathbf{y_2})) \end{aligned}$$

The main inferences for conjunctive queries are:

- Query answering. *An* answer *of a conjunctive query* $Q(\mathbf{x}, \mathbf{y})$ *w.r.t.* KB *is an assignment* θ *of individuals to distinguished variables, such that* $\pi(KB) \models \pi(Q(\mathbf{x}\theta, \mathbf{y}))$.
- Checking query containment. *A query* $Q_2(\mathbf{x}, \mathbf{y_2})$ *is contained in a query* $Q_1(\mathbf{x}, \mathbf{y_1})$ *w.r.t.* KB, *if* $\pi(KB) \models \pi(Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1}))$.

We now define the notion of rules and combined knowledge bases extended with rules.

Definition 4 (Rules) *A* rule *over a* $\mathcal{SHOIN}(\mathbf{D})$ *knowledge base* KB *has the form* $H \leftarrow Q(\mathbf{x}, \mathbf{y})$ *where* H *is an atom and* $Q(\mathbf{x}, \mathbf{y})$ *a query over* KB. *As usual, we assume rules to be* safe, *i.e. that each variable from* H *occurs in* $\mathbf{x}$ *as well. A rule is* DL-safe *if and only if* $Q(\mathbf{x}, \mathbf{y})$ *is DL-safe. We extend the operator* π *to translate rules into first-order formulas as follows:*

$$\pi(H \leftarrow Q(\mathbf{x}, \mathbf{y})) = \forall \mathbf{x} : \pi(H) \leftarrow \pi(Q(\mathbf{x}, \mathbf{y}))$$

A program P *is a finite set of rules;* P *is DL-safe if all rules are DL-safe. A* combined knowledge base *is a pair* (KB, P); *we define* $\pi((KB, P)) = \pi(KB) \cup \pi(P)$. *The main inference in* (KB, P) *is* query answering, *i.e. deciding whether* $\pi((KB, P)) \models A$ *for a ground atom* A.

To simplify the presentation, in the above definition we assume that H is a single atom, and not a conjunction of atoms. This is without loss of generality: it is well-known that a rule of the form $A_1 \wedge \ldots \wedge A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ is equivalent to the set of rules $A_i \leftarrow B_1 \wedge \ldots \wedge B_m$, for $1 \leq i \leq n$.

Notice that without DL-safety, the above definition matches that of the SWRL rules [HPS04a]. Intuitively, DL-safety restricts the applicability of a query or a rule only to individuals explicitly named in a knowledge base KB. To automatically convert a non-DL-safe query into a DL-safe one, we assume a special non-DL-predicate $\mathcal{O}$ such that, for each individual α occurring in KB, it contains a fact $\mathcal{O}(\alpha)$. Then, a non-DL-safe conjunctive query $Q(\mathbf{x}, \mathbf{y})$ can be easily converted into a DL-safe query by appending to it an atom of the form $\mathcal{O}(z)$, for each variable z occurring only in a DL-atom of $Q(\mathbf{x}, \mathbf{y})$. For an in-depth discussion about the consequences that this transformation has on the semantics, please refer to [MSS04].

6.3 A Mapping System for OWL DL

Based on the definitions from [Len02], we now introduce the notion of an OWL DL mapping system. The components of this mapping system are the source ontology, the target ontology, and the mapping between the two.

Definition 5 (OWL DL Mapping System) *An OWL DL mapping system $\mathcal{MS}$ is a triple $(\mathcal{S}, \mathcal{T}, \mathcal{M})$, where*

- *$\mathcal{S}$ is the* source OWL DL ontology,
- *$\mathcal{T}$ is the* target OWL DL ontology,
- *$\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{T}$, i.e. a set of assertions $q_S \rightsquigarrow q_T$, where q_S and q_T are conjunctive queries over $\mathcal{S}$ and $\mathcal{T}$, respectively, with the same set of distinguished variables $\mathbf{x}$, and $\rightsquigarrow \in \{\sqsubseteq, \sqsupseteq, \equiv\}$.*

An assertion $q_S \sqsubseteq q_T$ is called a sound *mapping, requiring that q_S is contained by q_T w.r.t. $S \cup \mathcal{T}$; an assertion $q_S \sqsupseteq q_T$ is called a* complete *mapping, requiring that q_T is contained by q_S w.r.t. $\mathcal{S} \cup \mathcal{T}$; and an assertion $q_S \equiv q_T$ is called an* exact *mapping, requiring it to be sound and complete.*

A sound mapping $q_S \sqsubseteq q_T$ is equivalent to an axiom $\forall \mathbf{x} : q_T(\mathbf{x}, \mathbf{y}_T) \leftarrow q_S(\mathbf{x}, \mathbf{y}_S)$, while a complete mapping $q_T \sqsubseteq q_S$ is equivalent to an axiom $\forall \mathbf{x} : q_S(\mathbf{x}, \mathbf{y}_S) \leftarrow q_T(\mathbf{x}, \mathbf{y}_T)$. We call these assertions *general implication mappings* to distinguish them from special types of mappings that we define later.

The generality of the above definition captures a broad class of approaches for mapping systems. Let us discuss the expressiveness in terms of the ontology language, the query language and the assertions. The source and target ontology are $\mathcal{SHOIN}(\mathbf{D})$ ontologies, i.e. logical theories that can have multiple models. In contrast, mapping systems in databases typically rely on simple relational schemas to describe the source and

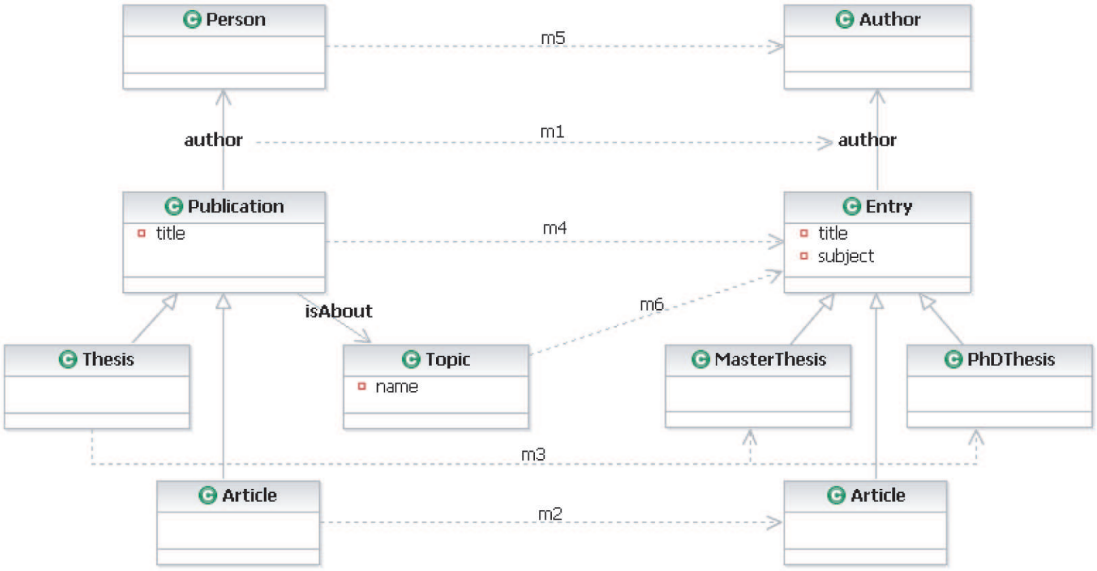


Figure 6.1: Example Mapping

target, and each source is assumed to be one database (with a single model). The expressiveness of conjunctive queries corresponds to that of the well-known select-project-join queries in relational databases. Two typical approaches to specify mappings are the *global-as-view* (GAV) approach, where elements of the target are described in terms of queries over source, and the *local-as-view* (LAV) approach, where elements of the source are described in terms of queries over target. Our mapping system subsumes the approaches of GAV, LAV. In fact, it corresponds to the GLAV approach, which is more expressive than GAV and LAV combined [FLM99]. We now present an example of a mapping system. We will continue the example in the subsequent parts of the section for illustration of the rather formal definitions.

Example 7 *Let us assume that we need to establish semantic correspondences between two heterogeneous ontologies modeling the bibliography domain. Table 6.2 shows the definition of the source ontology $\mathcal{S}$, and the target ontology $\mathcal{T}$. Figure 6.1 shows a visualization of the two ontologies along with the correspondences between them. The corresponding mappings $\mathcal{M}$ are shown in Table 6.3 in terms of our mappings system.*

We now define the semantics of the mapping system by translation into first-order logic, based on our definitions from Section 6.2:

Definition 6 (Mapping System Semantics) *For a mapping system $\mathcal{MS} = (\mathcal{S}, \mathcal{T}, \mathcal{M})$, let*

$$\pi(\mathcal{MS}) = \pi(\mathcal{S}) \cup \pi(\mathcal{T}) \cup \pi(\mathcal{M}).$$

Table 6.2: Source Ontology $\mathcal{S}$ and Target Ontology $\mathcal{T}$

Source Ontology	Target Ontology
$Person \sqsubseteq \top$	$Author \sqsubseteq \top$
$Publication \sqsubseteq \top$	$Entry \sqsubseteq \top$
$Article \sqsubseteq Publication$	$Article \sqsubseteq Entry$
$Thesis \sqsubseteq Publication$	$MasterThesis \sqsubseteq Entry$
	$PhDThesis \sqsubseteq Entry$
$Topic \sqsubseteq \top$	
$Person \sqsubseteq \forall name.String$	$Author \sqsubseteq \forall name.String$
$Topic \sqsubseteq \forall name.String$	
$\top \sqsubseteq \forall author.Person$	$\top \sqsubseteq \forall author.Author$
$Publication \sqsubseteq \forall title.String$	$Entry \sqsubseteq \forall title.String$
$Publication \sqsubseteq \forall isAbout.Topic$	$Entry \sqsubseteq \forall subject.String$

The main inference for $\mathcal{MS}$ is computing answers of $Q(\mathbf{x}, \mathbf{y})$ w.r.t. $\mathcal{MS}$, for $Q(\mathbf{x}, \mathbf{y})$ a conjunctive query.

To understand the intuition of computing answers, we briefly recall the the semantics of query answering as defined in Definition 3: An answer of a conjunctive query $Q(\mathbf{x}, \mathbf{y})$ w.r.t. KB is an assignment θ of individuals to distinguished variables, such that $\pi(KB) \models \pi(Q(\mathbf{x}\theta, \mathbf{y}))$. Thus, the intuitive reading of this semantics is that an answer of a query needs to be entailed by the source ontology $\mathcal{S}$, the target ontology $\mathcal{T}$ and the mappings $\mathcal{M}$. This semantics is equivalent to the usual model theoretic semantics (e.g. in [CGL01]) based on local and global models, where a query answer must be an answer in *every* global model.

Query answering in such a mapping system of this general form is undecidable and requires a theorem prover. In the following we introduce special types of mappings that lead to decidable query answering and for which practical query answering algorithms exist.

6.3.1 Full Implication Mappings

The first class of mappings captures the mappings that can be directly expressed in OWL DL. This is the case if q_s and q_t are of the form $P_s(\mathbf{x})$ and $P_t(\mathbf{x})$, where P_s and P_t are DL predicates:

Role Mappings. If q_s and q_t are of the form $P_s(x_1, x_2)$ and $P_t(x_1, x_2)$, with P_s and P_t are abstract or concrete roles, the mapping corresponds to the equivalent role inclusion axiom.

Concept Mappings. If q_s and q_t are of the form $P_s(x)$ and $P_t(x)$ and P_s, P_t are DL concepts, the mapping corresponds to the equivalent concept inclusion axiom.

Table 6.3: Mapping $\mathcal{M}$

Correspondences
$Q_{s,1}(x,y)$: *s:author*(x,y) $Q_{t,1}(x,y)$: *t:author*(x,y) $m_1 : Q_{s,1} \sqsubseteq Q_{t,1}$
$Q_{s,2}(x)$: *s:Article*(x) $Q_{t,2}(x)$: *t:Article*(x) $m_2 : Q_{s,2} \sqsubseteq Q_{t,2}$
$Q_{t,3}(x)$: *s:Thesis*(x) $Q_{s,3}(x)$: (*t:MasterThesis* $\sqcup$ *t:PhDThesis*)(x) $m_3 : Q_{s,3} \sqsubseteq Q_{t,3}$
$Q_{s,4}(x,t)$: *s:Publication*$(x) \wedge$ *s:title*(x,t) $Q_{t,4}(x,t)$: *t:Entry*$(x) \wedge$ *t:title*(x,t) $m_1 : Q_{s,4} \sqsubseteq Q_{t,4}$
$Q_{s,5}(x)$: *s:Person*$(x) \wedge$ *s:author*(y,x) $Q_{t,5}(x)$: *t:Author*(x) $m_5 : Q_{s,5} \sqsubseteq Q_{t,5}$
$Q_{s,6}(x,z)$: *s:Publication*$(x) \wedge$ *s:isAbout*$(x,y) \wedge$ *s:name*(y,z) $Q_{t,6}(x,z)$: *t:Entry*$(x) \wedge$ *t:subject*(x,z) $m_6 : Q_{s,6} \sqsubseteq Q_{t,6}$

Example 7 (continued) *Mapping m_1 simply maps the author property of the source ontology to that of the target ontology. It can be expressed with a simple role mapping:*

$$\textit{s:author} \sqsubseteq \textit{t:author}$$

Mapping m_2 maps the articles in the source ontology to articles in the target ontology, which can be expressed with a simple concept mapping:

$$\textit{s:Article} \sqsubseteq \textit{t:Article}$$

Mappings m_3 demonstrates the use of complex concepts in a concept mapping; it maps the concept Thesis in the source ontology to the union of the concepts PhDThesis and MasterThesis:

$$\textit{s:Thesis} \sqsubseteq (\textit{t:MasterThesis} \sqcup \textit{t:PhDThesis})$$

It shows that because of the expressiveness of the ontology language, we are able to express disjunctions in mappings (despite the fact that the query language only allows conjunctive queries).

6.3.2 Restricted Implication Mappings

It is well-known that query answering for general implication mappings is undecidable due to the unrestricted use of non-distinguished (i.e. existentially bound) variables in either q_S or q_T. In the following, we define restrictions that reduce the expressivity of the mappings, but provide for a decidable query answering procedure.

DL-safe Mappings. Let us consider a sound mapping $q_S \sqsubseteq q_T$[1] with the assertion $\forall \mathbf{x} : q_T(\mathbf{x}, \mathbf{y}_T) \leftarrow q_S(\mathbf{x}, \mathbf{y}_S)$. In order to avoid introducing new objects in the interpretation domain, we disallow the use of non-distinguished variables in the query q_T, i.e. restrict the assertions to the form $\forall \mathbf{x} : q_T(\mathbf{x}) \leftarrow q_S(\mathbf{x}, \mathbf{y}_S)$. Please note that these assertions directly corresponds to SWRL rules. Analogously to safe rules, we call these mappings safe mappings. Query answering with such mappings is still undecidable in the general case. Therefore, we require the query q_S to be DL-safe (c.f. Definition 3), thus limiting the applicability of the rules to known individuals. Thus obtained mappings correspond to (one or more) DL-safe rules from Definition 4, for which efficient algorithms for query answering are known [MSS04].

Example 7 (continued) *Mapping m_4 maps publications from the source ontology along with their title to the corresponding entries of the target ontology. The sound mapping is expressed via the assertion*[2]*:*

$$\forall x, y : \textit{t:Entry}(x) \wedge \textit{t:title}(x, y) \leftarrow \\ \textit{s:Publication}(x) \wedge \textit{s:title}(x, y)$$

This general implication mapping contains no non-distinguished variable in $Q_{t,4}$, so it can be expressed in a SWRL rule. However, the mapping is not DL-safe, as both x and y do not occur in non-DL predicates in $Q_{s,4}$. The mapping can be made DL-safe (as explained previously) by binding these variables with the special non-DL predicate $\mathcal{O}$ to individuals that actually occur in the source ontology:

$$\forall x, y : \textit{t:Entry}(x) \wedge \textit{t:title}(x, y) \leftarrow \\ \textit{s:Publication}(x) \wedge \textit{s:title}(x, y) \wedge \mathcal{O}(x) \wedge \mathcal{O}(y)$$

Mappings with Tree-like Query Parts. The restrictions introduced by DL-safety may appear rather strong. In the following we show how to relax the above restriction for a certain class of so-called tree-like queries. Using the query roll-up technique from [HT02], we can eliminate non-distinguished variables by reducing a tree-like part of a query to a concept, without loosing semantic consequences.

Definition 7 (Tree-Like Query Parts) *For a set of unary and binary literals S, the* coincidence graph *of S is a directed graph with the following structure:*

- *Each variable from S is associated with a unique node.*
- *Each occurrence of a constant in S is associated with a unique node, i.e. occurrences of the same constant are associated with distinct nodes.*

[1] For a complete mapping $q_S \sqsupseteq q_T$, the situation is analogous, with the roles of q_S and q_T reversed.

[2] In the mappings we use the namespace prefixes *s:* and *t:* to denote elements of the source and target ontology, respectively.

- *For each literal $C(s) \in S$, the node s is labeled with C.*
- *For each literal $R(s,t) \in S$, the nodes s and t are connected with a directed arc labeled R.*

The subset Γ of DL-atoms of a conjunctive query $Q(\mathbf{x},\mathbf{y})$ is called a tree-like part *of $Q(\mathbf{x},\mathbf{y})$ with a root s if*

- *no variable from Γ occurs in $Q(\mathbf{x},\mathbf{y}) \setminus \Gamma$,*
- *the coincidence graph of Γ is a connected tree with a root s,*
- *all nodes apart from s are non-distinguished variables of $Q(\mathbf{x},\mathbf{y})$.*

For details of the roll-up technique for tree-like queries, please refer to [HT02]; here we explain this technique on the following conjunctive query:

$$\exists y, z, w : R(x,y) \wedge A(y) \wedge S(y,z) \wedge B(z) \wedge T(y,w) \wedge C(w)$$

Since the entire query is tree-like with the root x, the existential quantifiers over z and w can be moved to the atoms where z and w first occur, yielding

$$\exists y : R(x,y) \wedge A(y) \wedge \begin{array}{l}[\exists z : S(y,z) \wedge B(z)] \wedge \\ [\exists w : T(y,w) \wedge C(w)]\end{array}$$

which is obviously equivalent to

$$\exists y : R(x,y) \wedge A(y) \wedge \exists S.B(y) \wedge \exists T.C(y).$$

Now the same procedure can be applied again, yielding the formula

$$\exists R.[A \wedge \exists S.B \wedge \exists T.C](x).$$

By these transformations we have eliminated the non-distinguished variables, i.e. we have "pushed them into description logic."

Example 7 (continued) *Mapping m_5 maps persons that are authors of a publication in the source ontology to authors in the target ontology. For the query $Q_{s,5}$ we can apply the query roll-up technique. $Q_{s,5}(x)$: s:Person$(x) \wedge$ s:author(y,x) is a tree-like query with the distinguished variable x as its root. It can thus be rolled-up to the semantically equivalent query $Q_{s,5}(x)$: $(\exists$s:author$^-$.s:Person$)(x)$. We can therefore express the mapping as a concept mapping:*

$$\exists \text{s:author}^-.\text{s:Person} \sqsubseteq \text{t:Author}$$

Let us compare this mapping with a DL-safe mapping obtained without query roll-up:

$$\forall x : \textit{t:Author}(x) \leftarrow \\ \textit{s:Person}(x) \wedge \textit{s:author}(y, x) \wedge \mathcal{O}(x) \wedge \mathcal{O}(y)$$

With the latter assertion, only those persons whose publications are explicitly named in the source ontology will be mapped to authors of the target ontology, whereas the prior mapping only requires a publication to exist, but it does not require it to be explicitly named. To illustrate the difference, consider the following ABox of the source ontology as example:

$$\textit{s:Person}(peter),\ \textit{s:Person}(boris), \\ (\exists \textit{s:author}^{-})(peter),\ \textit{s:author}(pub, boris)$$

A query against the target ontology $q_t(x) : \textit{t:Author}(x)$ *would return both individuals* $peter$ *and* $boris$ *as result for the mapping obtained with query roll-up, as for both individuals authored publications are known to exist. The same query, but evaluated with the latter mapping, would return only the individual* $boris$ *as query result, as the the publication of which* $peter$ *is an author, is not explicitly named in the ontology.*

Finally, mapping m_6 *maps the topic classification of the publications. Please note that the topics in the source ontology are modeled as a separate concept, whereas in the target ontology the entries carry the name of the topic as a property: The name of the topic in the source ontology maps to the subject of the entry in the target ontology. The mapping can be expressed with the following assertion:*

$$\forall x, z : \textit{t:Entry}(x) \wedge \textit{t:subject}(x, z) \leftarrow \\ \textit{s:Publication}(x) \wedge \textit{s:isAbout}(x, y) \wedge \textit{s:name}(y, z)$$

This mapping is again not DL-safe. Also, neither $Q_{s,6}$ *nor* $Q_{t,6}$ *are tree-like queries, so query roll-up can not be applied. To make the mapping DL-safe, we again require the variables to be bound to explicitly named individuals:*

$$\forall x, z : \textit{t:Entry}(x) \wedge \textit{t:subject}(x, z) \leftarrow \\ \textit{s:Publication}(x) \wedge \textit{s:isAbout}(x, y) \wedge \textit{s:name}(y, z) \\ \wedge \mathcal{O}(x) \wedge \mathcal{O}(y) \wedge \mathcal{O}(z)$$

6.4 Query Answering in an Ontology Integration System

In this section we show how to use an OWL DL mapping system from Section 6.3 for query answering in an *ontology integration system*, whose main task is to provide integrated access to a set of information sources, each expressed with a local source ontology. The integration is realized via a mediated target ontology through which we can query the local ontologies.

The algorithm is based on the correspondence between description logics and disjunctive datalog from [HMS04]. Given a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, this algorithm produces a positive disjunctive datalog program $\mathsf{DD}(KB)$ which entails exactly the same set of ground facts as KB, i.e. $KB \models A$ if and only if $\mathsf{DD}(KB) \models A$, for A a ground fact. Thus, query answering in KB is reduced to query answering in $\mathsf{DD}(KB)$, which can be performed efficiently using the techniques of (disjunctive) deductive databases. For example, the magic sets transformation [BR87] or various statistics-based join-order optimizations can be applied to $\mathsf{DD}(KB)$ to optimize query answering.

Due to some technical particularities, this algorithm requires KB to be a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base; all OWL DL constructs apart from nominals are supported. Another restriction is that the ground fact A is not allowed to contain complex roles (i.e. roles with transitive subroles). This is due to the approach used to handle transitivity axioms; for details, refer to [HMS04].

In [MSS04] it was shown that the above algorithm can be used to answer queries in a combined knowledge base (KB, P), where P is a DL-safe program: $(KB, P) \models A$ if and only if $\mathsf{DD}(KB) \cup P \models A$, for A a ground atom. Assuming a bound on the arity of the predicates in P, query answering can be performed in time exponential in the size of KB and P. Furthermore, as shown in [HMS05], the data complexity of these algorithms (i.e. the complexity assuming the size of the schema is fixed) is NP-complete, or even P-complete if disjunctions are not used.

For a set of local source ontologies $\mathcal{S}_1, \ldots, \mathcal{S}_n$, a target ontology $\mathcal{T}$ and corresponding mapping systems $\mathcal{MS}_1, \ldots, \mathcal{MS}_n$ with $\mathcal{MS}_i = (\mathcal{S}_i, \mathcal{T}, \mathcal{M}_i)$, an *ontology integration system* $\mathcal{IS}$ is again a mapping system $(\mathcal{S}, \mathcal{T}, \mathcal{M})$ with $\mathcal{S} = \bigcup_{i \in \{1 \ldots n\}} \mathcal{S}_i$ and $\mathcal{M} = \bigcup_{i \in \{1 \ldots n\}} \mathcal{M}_i$. The main inference task for $\mathcal{IS}$ is to compute answers of $Q(\mathbf{x}, \mathbf{y})$ w.r.t. $\mathcal{S} \cup \mathcal{T} \cup \mathcal{M}$, for $Q(\mathbf{x}, \mathbf{y})$ a conjunctive query over $\mathcal{T}$. Please note that because of the absence of a global ontology, this form of ontology integration system can be directly applied to decentralized integration: Assume we have a set of autonomous nodes, each relying on a local ontology, and a set of mapping systems that relate the local ontology to those of other nodes. An ontology integration system $\mathcal{IS} = (\mathcal{S}, \mathcal{T}, \mathcal{M})$ can easily be constructed for each individual node, where $\mathcal{S}$ consists of the ontologies of the remote node to be integrated, $\mathcal{T}$ is the ontology of local node, and $\mathcal{M}$ consists of the individual mappings systems describing the correspondences between the local ontology with remote ontologies.

Algorithm 1 shows how to compute answers to a conjunctive query $Q(\mathbf{x}, \mathbf{y})$ in an ontology integration system $\mathcal{IS}$. It is based on the algorithm for reducing description logics to disjunctive datalog outlined above, from which it inherits certain limitations: $\mathcal{IS}$ is required to be based on $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases, and the conjunctive query $Q(\mathbf{x}, \mathbf{y})$ and the queries in mappings are not allowed to contain transitive roles. The algorithm starts by eliminating non-distinguished variables from $Q(\mathbf{x}, \mathbf{y})$ and the mappings using the query roll-up technique. After roll-up, the obtained mappings and queries are

Algorithm 1 Algorithm for Answering Queries in an Ontology Integration System

Require: ontology integration system $\mathcal{IS}$ and a conjunctive query $Q(\mathbf{x}, \mathbf{y})$

1: Roll-up tree-like parts of $Q(\mathbf{x}, \mathbf{y})$
2: Roll-up tree-like parts of query mappings in $\mathcal{M}$
3: Stop if $Q(\mathbf{x}, \mathbf{y})$ or some mapping from $\mathcal{M}$ is not DL-safe
4: $\Gamma \leftarrow \mathsf{DD}(\mathcal{S} \cup \mathcal{T} \cup \mathcal{M})$
5: Compute the answer of $Q(\mathbf{x}, \mathbf{y})$ in Γ

required to be DL-safe, which is needed for decidable query answering. If this precondition is fulfilled, then the source ontology, target ontology and the mappings are converted into a disjunctive datalog program, and the original query is answered in the obtained program. By the results from [HMS04, MSS04], it is easy to see that the algorithm exactly computes the answer of $Q(\mathbf{x}, \mathbf{y})$ in $\mathcal{IS}$.

Let us contrast our approach to query answering with typical approaches in GAV and LAV data integration with respect to how queries against the target are reformulated to queries against the sources. In GAV systems, the problem reduces to simple view unfolding, as the reformulation is explicit in the mappings. In LAV, the problem requires more complex reasoning steps. In contrast, our approach does not require an explicit reformulation step. Instead, query answering here operates on a combined knowledge base consisting of source ontology, target ontology and mappings between them.

From the above definition, one might get the impression that the above algorithm requires that all source and target ontologies must be physically integrated into one mapping system in order to answer queries. This is, of course, not the case. More concretely, to compute $\mathsf{DD}(\mathcal{S} \cup \mathcal{T} \cup \mathcal{M})$, it is necessary to physically integrate the TBox and the RBox part of $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{M}$. Since the TBox and RBox are typically much smaller than the data, this does not pose practical problems. Accessing actual data sources (i.e. the ABoxes) is then governed by the chosen strategy for evaluating the disjunctive program.

6.5 Related Work

Our work is based on the formalization of a mapping system introduced in [Len02], which, because of its generality, subsumes a large class of mapping representations. Aspects that are not captured by this model include probabilistic or fuzzy mappings such as in the [BEE+04] and the notion of context as for example introduced in C-OWL [BGH+03]. [SSW05] provides a comparison of the approaches of Description Logics for information integration [CGL02], ontology integration systems [CGL01], C-OWL [BGH+03], and e-connections [GPS04] by an encoding in distributed first order logic (DFOL). Inspired by the dimensions considered in [SSW05], we briefly discuss and compare these formalisms for mapping representations in terms of (1) the type of

supported mappings, (2) the relationship of the mapped domains, (3) the arity of the mappings, (4) how inconsistencies are dealt with, and (5) their applicability and intended use.

Description Logics for Data Integration. The task of ontology integration as application of the mapping system is very related to that of data integration in databases. In [Len02] the author introduces a general framework for data integration and compares existing approaches to data integration (GAV, LAV) along this framework. He also discusses *query processing approaches* for GAV and LAV, as well as the topic of *inconsistencies between sources*, and *reasoning on queries*. In [Hal03] Halevy gives a status report on data integration, describing the recent progress on (i) schema mediation languages (LAV, GAV, GLAV), (ii) query answering algorithms (*view unfolding in GAV*, *answering queries using views in LAV*), (iii) query optimization, (iv) query execution , and (v) industry development. [CGL02] presents a description logic based approach to data integration. In this framework, the sources are described using views over a mediated schema (LAV). The Description Logic DLR is used to model the components of the data integration system. As query expressions, non-recursive datalog queries are allowed. Views are defined based on these query expressions. A query answering algorithm based on reduction of answering queries using views to unsatisfiability is presented.

Ontology Integration. The work on data integration has been extended and re-applied to ontology integration in [CGL01]. Here the authors follow the classical distinction between LAV and GAV approaches and outline query answering algorithms for these specific settings. In contrast to this work, query answering in our ontology integration system is not bound to these restricted forms of mappings. The main distinction between ontology integration and the existing approaches in data integration in databases is that in data integration one assumes that the sources basically are one *database*, whereas in ontology integration a source ontology is an *arbitrary logical theory*, which can have multiple models. Further, in data integration the languages to describe the sources and targets are typically very restricted (e.g. express the schemas as plain relations). Finally, the approaches are often limited to either LAV or GAV, whereas we do not make restrictions here.

C-OWL. In C-OWL [BGH+03], which is based on Distributed Description Logics (DDL, [BS02]), bridge axioms are used to semantically relate concepts and properties of different ontologies. A bridge axiom is an expression of one of the following forms:

$$C_i \xrightarrow{\sqsubseteq} C_j \; ; \; C_i \xrightarrow{\sqsupseteq} C_j \; ; \; a_i \longrightarrow a_j$$

where C_i, C_j are concepts, and a_i, a_j are individuals in the ontologies O_i, O_j, respectively. These bridge axioms establish directional subsumption relations between classes and correspondences between individuals in different ontologies. It is obvious from the type of primitives that C-OWL is adequate for relating different ontologies that model the

same or at least overlapping domains. It is a characteristic of C-OWL that inconsistencies do not propagate in general, which can be seen as a feature, if mutually inconsistent viewpoints (contexts) are to be modeled, but also as a flaw, if inconsistencies accross distributed ontologies cannot be detected.

E-connections. An E-Connection is a knowledge representation language defined as a combination of other logical formalisms. E-Connections were originally introduced in [KLWZ04] as a way to go beyond the expressivity of each of the component logics, while preserving the decidability of the reasoning services. An E-Connection language is a formalism, strictly more expressive than any of its component logics, and which is decidable, provided that each of its components is decidable. Hence, E-Connections provide a trade-off between the expressivity gained and the computational robustness of the combination. In [GPS04] E-Connections have been proposed as a language for defining and instantiating combinations of OWL DL ontologies, i.e. as a way of combining knowledge bases in one logical formalism, rather than combining different logical formalisms.

E-Connections introduce the notion of link properties that allow to link concepts in different ontologies without the need to import one ontology into the other, following a local model semantics, where there is no overlap in the domains. In other words, link properties allow to create classes in a certain ontology based on information from a different ontology, provided that the domains of the ontology are disjoint, both from a logical and a modeling perspective. E-Connections are therefore especially adequate for the purpose of modularization, but less suited for integrating ontologies that describe the same or largely overlapping domain in heterogeneous ways. In an E-Connection, for every pair of ontologies O_i and O_j, there exists a set of link properties e_{ij}, which are interpreted as binary relations between the domains of the ontologies O_i and O_j. A link from O_i to O_j can be used to define concepts in O_i in a way that is analogous to how roles are used to define concepts, e.g. $C_i \sqsubseteq \exists E.C_j$, with C_i and C_j concepts in O_i and O_j, and E a link property.

Summarizing one can say that ontology integration is useful for finding a globally shared model for one domain, contexts are useful for relating models of the same domain (or overlapping domains), where a globally consistent model cannot be found and instead local model semantics is suitable, and finally E-Connections are adequate for relating models of disjoint domains.

A final related problem is that of ontology matching and ontology alignment in the line of [DMDH02], [Noy04] and [Ehr06], where the goal is to manually or (semi-)automatically identify correspondences between ontologies, which finally result in mappings expressed in some formalism. Our work can be seen as complementary in the sense that the identified correspondences can be expressed in our mapping system and applied for tasks such as ontology integration.

6.6 Conclusions

In this chapter we have presented the formalization of a general mapping system for the integration of heterogeneous OWL DL ontologies. In this mapping system, the mappings between source and target ontology are specified as correspondences between conjunctive queries against the ontologies. The expressiveness of the mapping system is embodied in the ontology language ($\mathcal{SHOIN}(\mathbf{D})$), the supported query language (conjunctive queries), and the flexibility of assertions (GLAV approach). We have further identified a decidable fragment of mappings and a practical query answering algorithm for the task of ontology integration. All components of the mapping system can be fully expressed in OWL DL extended with DL-safe rules. It thus integrates well with current efforts for rule extensions to OWL. The presented algorithms are implemented based on the KAON2 ontology management system[3].

A problem not discussed in this chapter is the question of how the correspondences between elements of the ontologies are found in the first place. In the following chapter we present a similarity framework for ontologies, which can be used as basis for the discovery of mappings, as well as other tasks requiring similarity measures for ontologies.

[3] `http://kaon2.semanticweb.org/`

Chapter 7

Similarity for Ontologies

As we have seen in the previous chapters, integration of information is an important problem to address in distributed information systems in order to deal with the characteristic of heterogeneity. In the previous chapter, we have presented a mapping formalism for the integration of distributed OWL ontologies. We discussed how these mappings can be represented and used for reasoning tasks. We also discussed that an important, complementary task is the discovery of such mappings, i.e. the identification of correspondences between data sources. Currently, similarity based approaches are the most powerful technique for solving this problem, both for identifying correspondences on the schema level [Ehr06], but also for establishing correspondences between data values on the data level. Besides the problem of discovering correspondences, in distributed information systems there is a multitude of other tasks that rely on similarity based techniques. These include for example correlation measures between data objects as used in collaborative filtering and recommender systems, matching resources against requests as in capability-based matching and information retrieval. In many cases, similarity measures exploit the structure and semantics of an underlying ontology to improve their effectiveness. In this chapter we present a comprehensive framework for measuring similarity within and between ontologies as a formal and integrated foundation for the different tasks relying on ontology based similarity.

The similarity framework was initially presented in [EHSH05], where we based our work on an abstract ontology model [BEH+02] that – due to its generality – allows to abstract from the particularities of specific existing ontology languages. This generality comes at the cost of limited applicability for specific applications. Here, however, we present the similarity framework for the OWL ontology model. In doing so, we increase the applicability of the framework for applications adhering to the OWL ontology language, taking the specific syntax and semantics of the underlying description logic into account.

In Section 7.1 we illustrate the use of the similarity framework within the Bibster system to demonstrate the variety of usages of ontology-based similarity and the result-

ing need for the similarity framework. In Section 7.2 then introduce a formal model of similarity in ontologies. The main characteristics of the similarity framework is its layered structure, which we present in detail in Section 7.3. We define three layers on which the similarity between two ontology elements can be measured: data layer, ontology layer, and context layer, that cope with the data representation, ontological meaning and contextual information about ontology elements, respectively. In that way, different layers consider different characteristics of elements which are then combined using an aggregation of the individual similarity measures. Moreover, in each layer corresponding background information is used in order to define the similarity more precisely. We present specific similarity measures in Section 7.4. After a discussion of related work in Section 7.5 we summarize the main ideas in Section 7.6.

Parts of this chapter are based prior publications. We initially presented the similarity framework in [EHSH05]. Applications of ontology-based similarity in the Bibster system have been described in [HSvH04], [HHSTS05], [HEHS04], and [HBE+04].

7.1 Applications of Similarity in the Bibster System

In this section we illustrate a variety of different applications of similarity in the bibliographic application scenario and the Bibster system. Ontology-based similarity measures are used to realize a number of functionalities in the Bibster system:

Duplicate detection Due to the distributed nature and potentially large size of the Peer-to-Peer network, the returned result set for a query might be large and contain duplicate answers. Furthermore, because of the heterogeneous and possibly even contradicting representation, such duplicates are often not exactly identical copies. Ontology based similarity functions allows us to effectively determine the similarity between the different answers and to remove apparent duplicate results based on the intuition that answers with a similarity above a defined threshold are duplicates. Instead of confronting the user with a list of all individual results, we are able to present query results grouped by semantic duplicates. The most important source of heterogeneity is that due to differences in spelling. In the definition of the similarity measures we can additional background information about the domain into account, e.g. that often the first names of authors are abbreviated.

Collaborative Ontology Evolution In Bibster we initially assumed both the application (SWRC) and domain (ACM) ontologies to be globally shared and static. This basically holds for the application ontology, but users want to adapt the domain ontology continuously to their needs. This is largely motivated by the sheer size of the ACM topic ontology which makes browsing, and therefore also querying and manual classification, difficult for users.

To be able to adapt the ontologies of the users to the changing needs, we recommend potentially relevant changes based on the changes that other users with similar contexts have performed. The basic idea is as follows: Assume that for a target ontology we know similar ontologies called *neighbors* for short, then we would like to spot patterns in similar ontologies that are absent in our target ontology and recommend them to the target ontology. Another wording of the same idea is that we would like to extract ontology change operations that applied to the target ontology increases the similarity with its neighbors. We here rely on methods from Collaborative Filtering, as we will explain in detail in Chapter 9. Collaborative Filtering relies on correlation measures between rated elements and/or users. In our application scenario, the rated elements are the elements of the ontology. As correlation measure, we here need a similarity function that takes the ratings of ontology elements as contextual information into account. We consider a membership rating which asserts that a user wants to have a particular element as part of his ontology or not. We obtain a usage-based rating by counting queries issued by the user and instances in his knowledge base that reference a given concept.

Peer Selection with Semantic Overlay Networks As already introduced in Section 5.3.3, in the Bibster system we apply the model of expertise based peer selection as proposed in [HSvH04]. Based on this model, peers advertise semantic descriptions of their expertise specified in terms of the ACM topic ontology. The knowledge about the expertise of other peers forms a semantic overlay network, in which peers with a similar expertise are clustered. That means, a semantic link between two peers is established, if their expertise is similar. To select an appropriate set of peers to forward a query to, we need to determine how closely the semantic content of a query that references an ACM topic matches the expertise of a peer. We thus need a similarity function to compare subjects with expertise descriptions and expertise descriptions with each other.

7.2 Definition of Similarity

In this section we introduce basic definitions of similarity, upon which we build our framework for similarity in ontologies. In order to do so, we first discuss the notion of a similarity. Generally, similarity can be seen as a special form of relatedness [Res95]: Objects – or elements of some domain – are considered similar if they have common characteristics or share features that are the same. [Lin98] has elaborated on this observation in three intuitions of similarity:

- **Intuition 1:** The similarity between two objects is related to their commonality. The more commonality they share, the more similar they are.
- **Intuition 2:** The similarity between two objects is related to the differences between them. The more differences they have, the less similar they are.

- **Intuition 3:** The maximum similarity between two objects is reached when the objects are identical.

There have been different attempts to formalize these intuitions mathematically using the notion of a similarity function, or similarity measure. [Ric92] has defined a similarity measure as follows:

Definition 8 (Similarity Measure) *A similarity measure is a real-valued function* $sim(x, y) : U^2 \rightarrow [0, 1]$ *on a set* U *– the universe of objects – measuring the degree of similarity between* x *and* y*.*

A similarity measure can often be defined inverse to a distance function, measuring the differences between two objects. A distance function, δ, is a scale that assigns to every pair of objects a non-negative number, called their distance, satisfying the following the following axioms:

1. Non-negativity: $\delta(x, y) \geq 0$
2. Identity of Indiscernibles: $\delta(x, y) = 0 \iff x = y$
3. Symmetry: $\delta(x, y) = \delta(y, x)$
4. Triangle Inequality: $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$

However, this geometric approach is not unconditionally adequate for all similarity measures, as shown in [Tve03]. In particular, the triangle inequality does not always hold. Informally stated, the triangle inequality implies that if x is similar to y, and y is similar to z, then x and z cannot be very dissimilar from each other. [Tve03] mentions a simple counter example: While Jamaica is similar to Cuba (because of geographical proximity) and Cuba is similar to Russia (because of political proximity), Jamaica and Russia are not similar at all.

Therefore, we do not generally demand all the (analogous) properties of a distance function, as this would disqualify a number of useful measures for which these properties do not hold. However, it is generally agreed that sim ought to be maximal for identical objects (according to Intuition 3) and symmetric, i.e.

$$\begin{array}{lll} \forall x, y \in U \text{ it holds} & 1.\ sim(x, x) = 1 & \text{(maximality)} \\ & 2.\ sim(x, y) = sim(y, x) & \text{(symmetry)} \end{array}$$

Please note that with maximality we do also not require $sim(x, y) = 1 \iff x = y$. The reason lies in the fact that we want to allow objects have a maximal similarity even if they have distinct identities. For example, in the OWL ontology language – which does not apply the Unique Name Assumption – the same objects can be referred to via different identifiers.

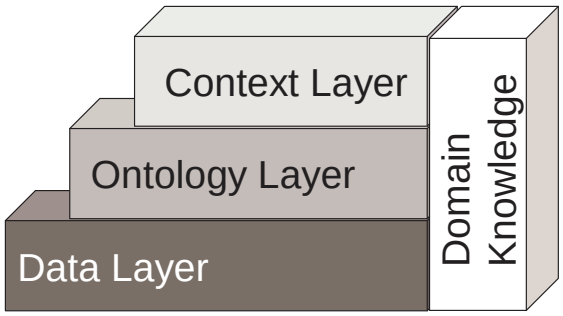


Figure 7.1: Layer Model of the Similarity Framework

Similarity for Ontologies

In the following we apply the generic definition of a similarity function to ontologies. Here, our universe of objects to compare consists of the elements of ontologies.

In particular, this means for our OWL ontology model that we consider as the universe the set of (atomic) concepts N_C, the sets of abstract and concrete roles N_{R_a} and N_{R_c}, and the sets of abstract and concrete individuals N_{I_a} and N_{I_c}, respectively. In the following, we simply write N to denote either of the ontology elements, if the distinction is not required to prevent ambiguity.

In the comparison however, we do not treat the elements as isolated entities, but consider the semantics of the ontology and the context they are being used in. These considerations are introduced in the following section.

7.3 Similarity Layer Model

Since an ontology represents a conceptualization of a domain, comparing two ontology elements goes far beyond the syntactic representation of these elements. Rather, it should take into account their relation to the real world entities they are referencing, i.e. their meaning, as well as their purpose in the real world, i.e. their usage. In order to achieve such a comprehensive comparison, we define our framework for similarity in three layers, as shown in Figure 7.1: Data-, Ontology-, and Context Layer. We further enhance these by an additional orthogonal dimension representing specific domain knowledge. A similar division into layers can be found for example in [MS02], where we find a *lexical* and *conceptual* layer.

7.3.1 Data Layer

On this first layer we compare elements simply based on their syntactic representation. This means for our OWL ontology model, when comparing concepts, roles and abstract individuals, the comparison is restricted to their symbolic name. In the case of concrete individuals, we consider their data values of the corresponding data types, such as integers and strings. Generally, we can define:

Definition 9 (Similarity on the Data Layer) *Let N denote the set of possible ontology elements, then a similarity function*

$$sim_{data} : (N)^2 \rightarrow [0, 1]$$

computes the similarity of elements based on the corresponding symbolic representation.

7.3.2 Ontology Layer

In the second layer, the ontology layer, we consider the semantics of the elements as defined by the ontology. This means we analyze how elements of the ontology are conceptually similar. Important conceptual structures to base similarity measures on include for example the subsumption hierarchy of the ontology, which allows to determine the taxonomic similarity based on the number of *is-a* edges separating two concepts. Besides intensional features we can also rely on the extensional dimension i.e. assess concepts to be the same, if their instances are similar. The similarity measures of the ontology layer can rely on similarity measures of the data layer to determine the basic similarities.

An important aspect is that we want to be able to compare ontology elements within single ontologies as well as across heterogeneous ontologies. Thus, in the similarity function we do not consider the elements as isolated entities, but considering the semantics of the ontology they are being used in.

We can thus define:

Definition 10 (Similarity on the Ontology Layer) *Let N denote the set of possible ontology elements and $\mathcal{O}$ the set of possible ontologies over N, then a similarity function*

$$sim_{ontology} : (N, \mathcal{O})^2 \rightarrow [0, 1]$$

computes the similarity of elements based on the semantics defined by their respective ontology.

7.3.3 Context Layer

On this layer we consider how the ontology elements are used in some external context. This implies that we use information external to the ontology itself. Our ontology model

so far describes the actual state of an ontology as an isolated entity. Now we need to consider contextual information about the ontology. The term "context" has many different connotations depending on the field it is being used in. In general, the context of an entity includes the circumstances and conditions which "surround" it. [Dey01] has defined context as: *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.* Applied to ontologies this means that we consider any information that is external to the ontology itself, but relevant for determining the similarity of the elements of the ontology. Typical examples of contextual information include – as we will later elaborate – usage information, provenance, etc.

To capture such contextual information about ontologies, we introduce the notion of *contextual annotations*, which are used to relate the context to the elements of the ontology:

Definition 11 *Let N denote the set of possible ontology elements and $\mathcal{X}$ be a suitable representation of a context space, then a* contextual annotation *is a partial function $r : N \rightarrow \mathcal{X}$.*

The context space $\mathcal{X}$ is kept as general as possible in this definition, to allow to attach essentially arbitrary information external to the ontology. We will later instantiate $\mathcal{X}$ for the specific contexts we consider.

We can now define:

Definition 12 (Similarity on the Context Layer) *Let N denote the set of possible ontology elements, $\mathcal{O}$ the set of possible ontologies over N and $\mathcal{R}$ the set of contextual annotations, then a similarity function*

$$sim_{context} : (N, \mathcal{O}, \mathcal{R})^2 \rightarrow [0, 1]$$

computes the similarity of ontology elements based on the contextual information about the ontology elements.

7.3.4 Domain Knowledge

Most of the aspects so far concerned aspects of similarity that are valid across domains. This means that corresponding similarity measures can be defined independently of the domain described by the ontology. However, specific domains have their own additional characteristics that may be exploited in determining similarity. The right part of Figure 7.1 therefore covers domain-specific aspects. As this domain-specific knowledge can be situated at any of the above layers, it is presented as a box across all of them. Just like we use general similarity features to compare ontologies we can also do so with domain specific features.

7.3.5 Similarity Aggregation

In general, the computation of the similarity of ontology elements will be based on a number of different individual similarity measures from the three layers that take different characteristics into consideration. For the computation of the overall similarity between two elements we use an aggregation function (referred to as amalgamation function in [EHSH05] and [Ric92]) that combines the results of the individual similarity functions of the layers described above:

Definition 13 (Aggregation of similarity functions) *Let $\mathcal{A}$ be a function $\mathcal{A} : [0,1]^n \rightarrow [0,1], n \geq 1$. We call $\mathcal{A}$ a* similarity aggregation function *if it satisfies the following axioms:*

1. $\mathcal{A}(0, \cdots, 0) = 0$ *and* $\mathcal{A}(1, \cdots, 1) = 1$
2. $\mathcal{A}$ *is monotone, i.e.* $x_1 \leq y_1, \cdots, x_n \leq y_n \implies \mathcal{A}(x_1, \cdots, x_n) \leq \mathcal{A}(y_1, \cdots, y_n)$
3. $\mathcal{A}$ *is symmetric, i.e. for every permutation* $i_1, \cdots, i_n$ *from* $1, \cdots, n$ *it holds:* $\mathcal{A}(x_1, \cdots, x_n) = \mathcal{A}(x_{i_1}, \cdots, x_{i_n})$

An aggregated similarity value $sim = \mathcal{A}(sim_1, \cdots, sim_n)$ is obtained by applying the aggregation function $\mathcal{A}$ to the individual similarities sim_i ($i \in \{1, \cdots, n\}$).

7.4 Specific Similarity Functions

After having presented a general framework for similarity, we now show specific measures which fit the framework. They are formally defined according to our ontology model. The list of measures is not intended to be exhaustive, but provides overview of important individual measures. This section contains both general similarity measures for ontologies and measures which fit only special domain ontologies. We also show approaches to aggregate the individual similarity measures. Most of the measures are not new, but rather constitute well established measures. The novelty rather constitutes in the description of these measures with our framework tailored to the OWL ontology model. We now describe specific measures applied in the framework grouped by the layers *Data Layer*, *Ontology Layer*, and *Context Layer*. We also show how we exploit *Domain Knowledge* in the individual layers.

7.4.1 Data Layer

On this layer we compare the symbolic representations of names of ontology elements, or data values in the case of concrete individuals, on a purely syntactic level. There exist several approaches to define the similarity on this level, a selection of which we present in the following.

Equality For some application scenarios it is appropriate to require equality of data values for elements to be considered similar:

$$sim_{equality}(n_1, n_2) := \begin{cases} 1, & \text{if } n_1 = n_2, \\ 0, & \text{otherwise} \end{cases} \tag{7.1}$$

This is without doubt the simplest similarity function satisfying the properties characterized in Definition 8. Such a definition is for example useful for data values that are used as uniquely identifying keys of elements.

Syntactic Similarity To cope with the heterogeneity of syntactic representations, for example due to differences in spelling, [Lev66] introduced a measure to compare two strings, the so called edit distance. For our purposes of similarity we rely on the syntactic similarity of [MS02] which is inverse to the edit distance measure:

$$sim_{syntactic}(v_1, v_2) := max(0, \frac{min(|v_1|, |v_2|) - ed(v_1, v_2)}{min(|v_1|, |v_2|)}) \tag{7.2}$$

The idea behind this measure is to take two strings and determine how many atomic actions are required to transform one string into the other one. Atomic actions are addition, deletion, and replacement of characters, but also moving their position. This measure is useful to handle spelling errors or mismatches in capitalization. Such discrepancies are frequently found in our application scenario of the Bibster system Consequently, we rely on the syntactic similarity function to detect these differences in the bibliographic metadata.

Further, domain specific background knowledge can be used to define more meaningful similarity measures. For example, for bibliographic metadata we know that attributes such as first and middle names are often abbreviated: In these cases it is possible to compare only the characters in front of the abbreviation dot.

Distance-Based Similarity for Numeric Values For numeric datatypes with a limited range (e.g. subsets of integer or double) it is advisable to use a similarity measure that assigns the similarity between two numerical values on the basis of their arithmetical difference. A generic example of such a distance-based similarity function is

$$sim_{diff}(v_1, v_2) := 1 - \left(\frac{|v_1 - v_2|}{V_{max} - V_{min}} \right)^{\gamma} \tag{7.3}$$

where $v_1, v_2 \in V = [V_{min}, \ldots, V_{max}] \subset \mathbb{R}$ for a numeric datatype V. The parameter $\gamma \in \mathbb{R}$ may be used to scale the influence of increasing differences between the values on the similarity.

7.4.2 Ontology Layer

We now turn to specific similarity functions that take the semantics of the ontology into account. We do so by stating the similarity functions in terms of reasoning problems. This allows us to talk about similarity in terms of facts entailed by the ontology, instead of in terms of explicitly asserted data structures as done in prior approaches, e.g. [MS02]. For example, when talking about equivalence of elements, we can test whether our ontology entails that two elements are equivalent.

Equivalence and Equality In the previous Equation 7.1 we relied on the identity of symbols to assert that elements with identical names are maximally similar, and dissimilar otherwise. Such a definition is inline with the unique name assumption, which however, does not hold in OWL ontologies. Taking the ontology into account, we can define the similarity for two concepts C_1 and C_2 in an ontology O as

$$sim_{equivalence}((C_1, O), (C_2, O)) := \begin{cases} 1, & \text{if } O \models C_1 \equiv C_2, \\ 0, & \text{otherwise} \end{cases} \tag{7.4}$$

A similar function can be defined for the equivalence of roles and equality of individuals. Please note that the function above is defined for the case that elements of the same ontology are compared. There are various ways to extend the definition to compare elements of different ontologies. In the simplest case, we could define:

$$sim_{equivalence}((C_1, O_1), (C_2, O_2)) := \begin{cases} 1, & \text{if } O_1 \cup O_2 \models C_1 \equiv C_2, \\ 0, & \text{otherwise} \end{cases} \tag{7.5}$$

In this case, we consider a global interpretation for both ontologies. It would also be possible to define the entailment in terms of local interpretations, e.g. in the style of C-OWL [BGH+03].

Taxonomic Similarity An important source of information for determining similarity is the concept hierarchy of an ontology. Computing the concept hierarchy for an ontology is a standard reasoning task for description logics based ontologies. The main intuition behind using the concept hierarchy is that concepts that are close in the hierarchy are similar.

There has been a long history of research related to taxonomic similarity that is directly relevant. The traditional edge based approach determines the distance length between nodes in the hierarchy. The shorter the path from one node to the other, the more similar they are. [LBM03] have compared different similarity measures and have shown that for measuring the similarity between two concepts C_1 and C_2 in a hierarchically structured semantic network (in this case the concept hierarchy) the following similarity measure yields the best results (evaluated against human ratings):

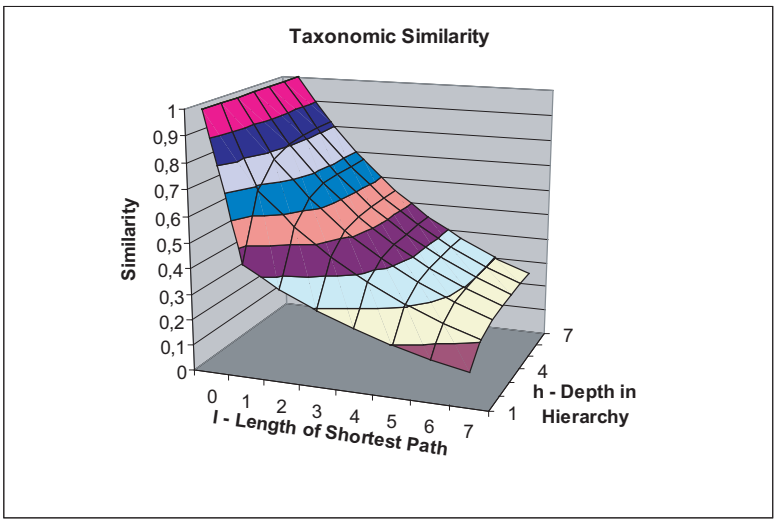


Figure 7.2: Taxonomic Similarity

$$sim_{taxonomic}((C_1, O), (C_2, O)) := \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, & \text{if } C_1 \neq C_2, \\ 1, & \text{otherwise} \end{cases} \tag{7.6}$$

$\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h of the deepest common super concept in the concept hierarchy of the ontology O, respectively. Again, for the case of comparing concepts from two different ontologies O_1 and O_2 it is possible to define the metric against the concept hierarchy of the combined ontology $O_1 \cup O_2$. The shortest path length is a metric for measuring the conceptual distance of C_1 and C_2. The intuition behind using the depth of the direct common super concept in the calculation is that concepts at upper layers of the concept hierarchy are more general and are semantically less similar than concepts at lower levels. The effect of path length l and depth h are shown in Figure 7.2.

Using the same intuitions, the taxonomic similarity can not only be applied to concept hierarchies, but also to individuals that are organized in a hierarchy using special roles, such as instances of a `Topic` concept that are organized using a `subtopic` relation.

In the Bibster system we rely on the taxonomic similarity in various places: In the process of expertise-based peer selection, we need to determine how closely the semantic content of a query that references an ACM topic matches the expertise of a peer. We here rely on the taxonomic similarity, which we apply to the ACM topic ontology to compare subjects with expertise descriptions and expertise descriptions with each other. Further,

we rely on the taxonomic similarity to compare the classification of publications against the ACM topic ontology in the process of duplicate detection.

Taxonomic Overlap Another measure that relies on the concept hierarchy is the taxonomic overlap as proposed by Maedche and Staab [MS02]. It compares concepts based on their intensional semantics as defined by the *semantic cotopy*. Here, the semantic cotopy of a concept C in an ontology O is the set of all super- and subconcepts of C:

$$SC(C, O) := \{C_i \mid O \models C \sqsubseteq C_i \vee O \models C_i \sqsubseteq C\}$$

While the original measure for taxonomic overlap in [MS02] was defined to determine the similarity of entire hierarchies, it can also be adopted to calculate the similarity of single concepts in heterogeneous ontologies:

$$sim_{cotopy}((C_1, O_1), (C_2, O_2)) := \frac{|SC(C_1, O_1) \sqcap SC(C_2, O_2)|}{|SC(C_1, O_1) \sqcup SC(C_2, O_2)|} \tag{7.7}$$

Extensional Concept Similarity In contrast to the intensionally defined taxonomic similarity, extensional concept similarity measures consider to which extent the concepts share individuals. Two concepts are similar, if their instances are similar. We here rely on the reasoning task of instance retrieval to determine the extension of a particular concept. We denote $C(x)$ as the extension of a concept as a shorthand for $\{x \mid O \models C(x)\}$.

$$sim_{extension}((C_1, O), (C_2, O)) := \frac{|C_1 \sqcap C_2(x)|}{|C_1 \sqcup C_2(x)|} \tag{7.8}$$

This function resembles the Jacquard Coefficient, which determines the fraction of overlapping elements in relation to the number of all elements in the two extensions. There are other functions to compute the similarity of sets of elements, including for example the Dice and cosine measure. As the individual entities have various and very different features, it is difficult to create a vector representing whole sets of individuals. Therefore we use a technique known from statistics as multidimensional scaling [CC94]. Multidimensional scaling encompasses methods which allow to gain insight in the underlying structure of relations between entities by providing a geometrical representation of these relations. Applied to determining the similarity of sets of ontology elements, we describe each element through a vector representing the similarity to any other element contained in the two sets. This can easily be done, as we rely on other measures which already did the calculation of similarity values $[0..1]$ between single elements. For both sets a representative vector can now be created by calculating an average vector over all elements. Finally we determine the cosine between the two set vectors through the scalar

product as the similarity value.

$$sim_{set}(E,F) := \frac{\sum_{e \in E} \vec{e}}{|E|} \cdot \frac{\sum_{f \in F} \vec{f}}{|F|} \tag{7.9}$$

with element set $E = \{e_1, e_2, \ldots\}$, $\vec{e} = (sim(e,e_1), sim(e,e_2), \ldots sim(e,f_1), \ldots)$; F and f are defined analogously. The intuition of this measure is that the scalar product will be large for similar vectors and small for orthogonal (dissimilar) vectors.

Role Similarity based on Domain and Range Restrictions We can define a similarity measure for roles based on their domain and range restrictions. We again rely on the subsumption hierarchy of an ontology O and compare the most specific concepts occurring as the domain or range in a role restriction. We define the domain $dom(R,O)$ as the most specific atomic concept C for which $O \models \forall R.\top \sqsubseteq C$. Specifically, with most specific concept C we mean that there may exist no other concept D that is subsumed by C with $O \models \forall R.\top \sqsubseteq D$. Analogously, we define the range $range(R,O)$ as the most specific atomic concept C for which $O \models \forall R^{-1}.\top \sqsubseteq C$. We can now reduce the comparison of roles based on their domain and range restrictions by reduction to a comparison of the corresponding concepts using some concept similarity measure:

$$sim_{dom}((R_1,O_1),(R_2,O_2)) := sim_{concept}(dom(R_1,O_1), dom(R_2,O_2)) \tag{7.10}$$

$$sim_{range}((R_1,O_1),(R_2,O_2)) := sim_{concept}(range(R_1,O_1), range(R_2,O_2)) \tag{7.11}$$

Similar definitions would be possible using existential role restrictions.

Concept Similarity of Individuals The concept hierarchy and the corresponding taxonomic similarity can also be used to define the similarity of individuals based on their most specific concepts. We here rely on the reasoning task of realization, i.e. the task of computing the most specific atomic concepts that are instantiated by a given individual, which can be reduced to a combination of retrieval and classification: For an individual a and an atomic concept C, C realizes a, if a is an instance of C and there is no atomic concept D, such that a is an instance of D, and C subsumes D. Correspondingly, the similarity of two individuals can be defined in terms of the similarity of the concepts they realize.

Individual Relation Similarity A further way of comparing individuals is based on considering how the individuals are related with other individuals via a given role R.

We denote $R(a, x)$ w.r.t. O as a query with x as a distinguished variable and a as an individual, i.e. a shorthand for $\{x \mid O \models R(a, x)\}$. Based on the answer set of such queries we can define:

$$sim_R((a_1, O_1), (a_2, O_2)) := sim_{set}(R(a_1, x), R(a_2, x)) \tag{7.12}$$

For example, in the Bibster system we use this function to compare two publications on the basis of the similarity of the sets of authors.

7.4.3 Context Layer

On the context layer, we extend the definition of similarity with the contextual information about ontology elements, in the sense that similar elements are used in a similar context.

Usage Context A very typical example of context information is the *usage context*. The intention of modeling usage context is to capture the users' behavioral patterns, which can in turn be used to assess the similarity based on the interaction with the ontology. We here use the contextual annotations to indicate the importance or relevance of particular elements. For example, if users interested in knowledge management (as obtained from the usage profile) are also interested in the topic Information Systems, one can derive the similarity between these two topics. Often such similarity calculations can be augmented with techniques from machine learning, in particular usage mining.

Generally, we can distinguish between *explicit* and *implicit* user feedback from usage information. We talk about explicit feedback if we allow that a user (i) can directly express how important a certain ontology element is for him, and that he (ii) can explicitly express negative ratings for ontology elements that he does not want to be part of his ontology.

We can obtain implicit feedback from log information that indirectly indicate the importance of ontology elements. For example, we can use an implicit usage context called $r^u : N \rightarrow \mathbb{N}$, which indicates the relevance of the elements based on how they have been used, e.g. counts the number of queries issued by the user and instances in his knowledge base that reference a given symbol name. This information is available in a wide range of application scenarios. Of course, in specific scenarios further information may be available and thus additional contextual annotations can be defined.

Based on the contextual annotations, it is now possible to define similarity functions for ontology elements based on the contextual annotations of the elements:

$$sim_{context}((n_1, O_1, r_{O_1}), (n_2, O_2, r_{O_2})) := sim(r_{O_1}(n_1), r_{O_2}(n_2)) \tag{7.13}$$

The definition of the specific function to compare the contextual annotations depends on the context space under consideration. For example, for the case of the usage context r^U,

where the context space is that of the natural numbers, we can rely on the corresponding similarity.

The definition of similarity of context can be extended to compare entire ontologies. We can choose a simple correlation measure (vector/cosine similarity) to compute similarities between ontologies of two users based on their contextual annotations of the ontology elements, assuming a shared space of elements N for both ontologies:

$$\text{sim}_r(O_1, O_2) := \frac{\sum_{s \in N} r_{O_1}(s)\, r_{O_2}(s)}{\sqrt{\sum_{s \in N} r_{O_1}(s)^2} \sqrt{\sum_{s \in N} r_{O_2}(s)^2}} \tag{7.14}$$

Such a correlation measure is useful for the case of comparing personal ontologies of users with their own usage context.

A very powerful tool for using usage context comes from information theory. The information theory-based method for semantic similarity was first proposed by Resnik [Res95]. He defines the similarity of two concepts as the maximum of the information content of the concept that subsumes them in the concept hierarchy. The information content of a concept depends on the probability of encountering an instance of a concept, where encountering an instance may refer to any form of usage of that concept. That is, the probability of a concept is determined by the frequency of occurrence of the concept and its subconcept. The information content is then defined as the negative log likelihood of the probability.

Assume we have a special contextual annotation p capturing the probability of encountering an instance of concept C, denoted with $p(C)$. Then the information content of concept C is quantified by following the notation of information theory as

$$IC(C) = -log(p(C))$$

As probability increases, the information content decreases, so the more abstract a concept, the lower its information content. The information content of the root concept is 0. This quantitative characterization of information provides a new way to measure semantic similarity. The more information two concepts share in common, the more similar they are, and the information shared by two concepts is indicated by the information content of the concepts that subsume them in the taxonomy. Formally, we can define: For two concepts C_1 and C_2 in an ontology O, the similarity measure is determined by the ratio between the information content of the subsumers in the concept hierarchy and the information content of the compared concepts:

$$sim_{IC}((C_1, O, p), (C_2, O, p)) = \max_{C_0 \in sub(C_1, C_2)} \frac{2 \times log(p(C_0))}{log(p(C_1)) + log(p(C_2))} \tag{7.15}$$

where $sub(C_1, C_2)$ is the set of concepts that subsume both C_1 and C_2 in O.

In a sense, this intuition is analogous to that of the taxonomic similarity, where an increasing depth in the hierarchy increases the similarity between two concepts.

7.4.4 Aggregation of Similarities

In the previous section we have listed a large number of possible measures—distributed over the three layers of our framework—to assess the similarity between certain elements. Without any doubt, that enumeration does not assert a claim on completeness. The crucial question arising is how to combine several similarity measures, e.g. when comparing elements based on a number of different characteristics.

Intending to aggregate several individual similarities as described in Section 7.4, one can think of various realizations of the aggregation function $\mathcal{A}$. However, it is plausible to claim that $\mathcal{A}$ is monotonous in each of its arguments and that it holds $\mathcal{A}(0,\ldots,0) = 0$ and $\mathcal{A}(1,\ldots,1) = 1$.

The probably most common approach to aggregate a number of individual similarities is to form the global similarity sim by assigning weights w_i to all involved individual similarities and compute sim as a weighted sum:

$$\mathcal{A}(sim_1,\ldots,sim_n) = \frac{\sum_{i=1}^{n} w_i \cdot sim_i}{\sum_{i=1}^{n} w_i} \tag{7.16}$$

The weighted average allows a very flexible definition of what similar means in a certain context. For example, for the duplicate detection in the Bibster system, we aggregate the individual similarities using a weighted sum, where the weights have been assigned manually on the basis of experiments. Here, the similarity based on the title has a high weight, and in order for two publications to be considered as duplicates, the global similarity needs to be close to 1.

However, recent results also show that the weights can also be effectively assigned using Machine Learning techniques [ESS05]. Alternatively, one might define $\mathcal{A}$ to select the minimum/maximum of its parameters or calculate any appropriate, application-specific combination of the local similarities. Another alternative is described in [JLQB06], where the authors propose an ordered weighting aggregation operator, which does not associate a weight is with a particular measure sim_i, but with with a particular ordered position. This allows to define aggregations based on *linguistic quantifiers*, such as *most* or *as many as possible* individual similarities.

7.5 Related Work

Similarity measures for ontological structures have been widely researched, e.g. in cognitive science, databases, software engineering, linguistics, and AI. Though this research covers many areas and application possibilities, most applications have restricted their attention to the determination of the similarity of the lexicon, concepts, and relations within one ontology.

The nearest to our approach of comparing elements across ontologies come [Bis92] and [WB99]. In [Bis92] the attention is restricted to the conceptual comparison level.

In contrast to our work the new concept is described in terms of the existing ontology. Furthermore, the author does not distinguish relations into taxonomic relations and other ones, thus ignoring the semantics of inheritance. [WB99] computes description compatibility in order to answer queries that are formulated with a conceptual structure that is different from the one of the information system. In contrast to our approach their measures depend to a very large extent on a shared ontology that mediates between locally extended ontologies. Their algorithm also seems less suited to evaluate similarities of sets of lexical entries, taxonomies, and other relations.

Much research on similarity has been carried out since the beginning of the 1980s in the area of database schema integration. Schema comparison analyzes and compares schema in order to determine correspondences and comes therefore near to our approach. The most relevant to our framework is the classification of schema matching approaches given in [RB01]. The authors distinguish three levels of abstraction. The highest level differs between schemata- and instance-based information. The second level distinguishes the similarity among elements and among structures. On the third level the calculation can be based on linguistic or information about a model's constraints. On the other hand our approach uses a conceptual decomposition: if the similarity of entities can be discovered on the data representation level (e.g. two strings are similar), then it can be expanded to the semantic level (e.g. if these strings are label for two concepts, then it can be an evidence that the concepts are similar) and finally this information can be propagated on the level of the usage of these concepts (e.g. if they are used similarly, then there is more evidence for their similarity). In that context our framework is more comprehensive, whereas all methods mentioned in [RB01] can be found in our framework. Moreover, we use background information about the given domain and not only "auxiliary" linguistic information (like synonyms, hypernyms) in all layers. The idea of measuring similarities between ontologies at different semiotic levels has also been applied by Maedche and Staab [MS02], who consider measures at the lexical and conceptual level, which roughly correspond to our data and ontology layer, respectively.

7.6 Conclusions

In this chapter we have presented a general framework for calculating similarity among elements in ontologies. We have extended the work of [EHSH05] and defined the framework in terms of the OWL ontology model. The main characteristic of the framework is its layered structure: We have defined three levels on which the similarity between two elements can be measured: data layer, ontology layer, and context layer, which cope with the data representation, ontological meaning, and contextual information about these elements, respectively. Our intention was not only to develop a collection of existing methods for measuring similarity, but rather to define a framework that will enable their effective systematization. For the individual layers we have further presented a number

of specific similarity measures. The measures on the data layer and ontology layer are directly defined in terms of the OWL ontology model. For the definition of similarity on the context layer we have additionally introduced the notion of contextual annotations. On the ontology layer we have defined specific measures based on standard description logic reasoning tasks, including computing concept hierarchies, instance retrieval, and realization of concepts. We have also shown how particular similarity functions are applied for different purposes in our motivating application scenario of the Bibster system. Throughout the work, we will refer to the similarity framework to explain these particular tasks in more detail.

Part III

Ontology Evolution

Chapter 8

Consistent Evolution of OWL Ontologies

In the preceding part of this work we dealt with the heterogeneity in distributed information systems and presented ontology-based information integration as a solution to the challenges arising from heterogeneity. In this part we turn to the second important characteristic, that of dynamics. Most of the work conducted so far in the field of ontologies has focused on ontology construction issues, which assumes that domain knowledge encapsulated in an ontology does not change over time. However, ontologies in real-world distributed information systems are typically not static entities, they change over time. These changes may be related to modifications in the application domain, incorporating additional functionality according to changes in the users' needs, organizing information in a better way, etc. In this part, we present methods for *ontology evolution* as solutions for managing changes to ontologies.

One of the major problems of evolving ontologies is the potential introduction of inconsistencies as a result of applying changes. Ontology evolution can be defined as the timely adaptation of an ontology to requested changes and the consistent management of these changes. It is not a trivial process, due to the variety of sources and consequences of changes, it thus cannot be left as manual work to the users of the information system. Therefore, explicit support for this process is required within distributed information systems. In this chapter we present an approach for the consistent evolution of OWL DL ontologies building on and extending the evolution process originally defined in [SMMS02]. We present a general overview of this process in Section 8.1. An important aspect in the evolution process is to guarantee the consistency of the ontology when changes occur, considering the semantics of the ontology change. A formalization of the semantics of change requires a definition of the ontology model together with its change operations, the consistency conditions and rules to enforce these conditions. We provide these formalizations in Section 8.2, where we define the notions of ontology change operations, the semantics of change, and the consistency of an ontology. We introduce

resolution strategies to ensure that consistency is maintained as the ontology evolves. We further define methods for detecting and resolving inconsistencies in an OWL ontology after the application of a change. Finally, as for some changes there may be several different consistent states of the ontology, we define resolution strategies allowing the user to control the evolution. We exemplarily present resolution strategies to detect and resolve structural inconsistency, logical inconsistency and user-defined inconsistency in Sections 8.3, 8.4, and 8.5, respectively.

The approach of consistent ontology evolution imposes certain requirements with respect to its applicability. For examples, it requires that the ontology is consistent in the first place and that changes to the ontology can be controlled. In certain application scenarios, these requirements may not hold, and consequently, other means for dealing with inconsistencies in changing ontologies may be required. In Section 8.6 we therefore compare alternative approaches to deal with inconsistencies in changing ontologies in terms of a common framework. Finally, we discuss related work in Section 8.7 and summarize the main findings in Section 8.8.

8.1 Evolution process

Ontology evolution can be defined as the timely adaptation of an ontology and consistent management of changes. The complexity of ontology evolution increases as ontologies grow in size, so a structured ontology evolution process is required. We follow the process described in [Sto04a]. The process starts with *capturing changes* either from explicit requirements or from the result of change discovery methods. Next, in the *change representation* phase, changes are represented formally and explicitly. The *semantics of change* phase prevents inconsistencies by computing additional changes that guarantee the transition of the ontology into a consistent state. In the *change propagation* phase all dependent artifacts (ontology instances on the Web, dependent ontologies and application programs using the changed ontology) are updated. During the *change implementation* phase required and induced changes are applied to the ontology in a transactional manner. In the *change validation* phase the user evaluates the results and restarts the cycle if necessary.

In this chapter we focus on the semantics of change phase. Its role is to enable the resolution of a given ontology change in a systematic manner by ensuring the consistency of the whole ontology. It is realized through two steps:

- *Inconsistency Detection*: This step is responsible for checking the consistency of an ontology with the respect to the ontology consistency definition. Its goal is to find ”parts” in the ontology that do not meet consistency conditions;

- *Change Generation*: This step is responsible for ensuring the consistency of the ontology by generating additional changes that resolve detected inconsistencies.

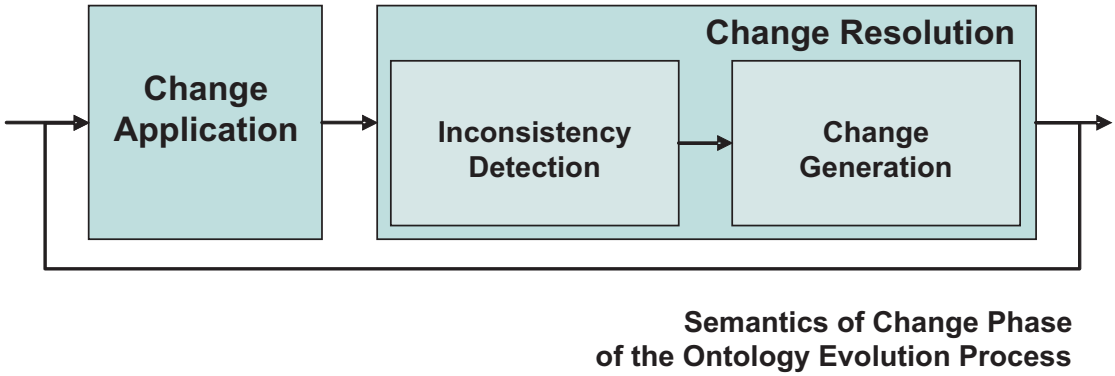


Figure 8.1: Semantics of Change Phase

The semantics of change phase of the ontology evolution process is shown in Figure 8.1. Changes are applied to an ontology in a consistent state (c.f. Change Application in Figure 8.1), and after all the changes are performed, the ontology must remain consistent (c.f. Change Resolution in Figure 8.1). This is done by finding inconsistencies in the ontology and completing required changes with additional changes, which guarantee the consistency. Indeed, the updated ontology is not defined directly by applying a requested change. Instead, it is defined as an ontology that satisfies the user's requirement for a change and is at the same time a consistent ontology.

In our work we specifically consider the semantics of change phase for OWL DL ontologies. Ontology consistency in general is defined as a set of conditions that must hold for every ontology [Sto04a]. Here, we have to distinguish various notions of consistency:

- *Structural Consistency:* First, we have to consider the structural consistency, which ensures that the ontology obeys the constraints of the ontology language with respect to how the constructs of the ontology language are used.

- *Logical Consistency:* Then, we need to consider the formal semantics of the ontology: Viewing the ontology as a logical theory, we consider an ontology as logically consistent if it is satisfiable, meaning that it does not contain contradicting information.

- *User-defined Consistency*: Finally, there may be definitions of consistency that are not captured by the underlying ontology language itself, but rather given by some application or usage context. The conditions are explicitly defined by the user and they must be met in order for the ontology to be considered consistent.

We note that most of the existing evolution systems (including the schema evolution systems as well) consider only the structural consistency. The role of an ontology

evolution system is not only to find inconsistencies in an ontology and to alert an ontology engineer about them. Helping ontology engineers notice the inconsistencies only partially addresses the issue. Ideally, an ontology evolution system should be able to support ontology engineers in resolving problems at least by making suggestions how to do that.

Moreover, an inconsistency may be resolved in many ways. In order to help the user to control and customize this process, we have introduced the so-called resolution strategies. Resolution strategies are developed as a method of "finding" a consistent ontology that meets the needs of the ontology engineer. A resolution strategy is the policy for the evolution with respect to his requirements. It unambiguously defines the way in which a change will be resolved, i.e. which additional changes will be generated.

In the remainder of this chapter we formally define different types of consistency and elaborate on how corresponding inconsistencies can be detected and resolved.

8.2 Change Representation and Semantics of Change

The goal of ontology evolution is to guarantee the correct semantics of ontology changes, i.e. ensuring that they produce an ontology conforming to a set of consistency conditions. The set of ontology change operations – and thus the consistency conditions – depends heavily on the underlying ontology model. Most existing work on ontology evolution builds on frame-like or object models, centered around classes, properties, etc. However, as in this work we focus on the evolution of OWL DL ontologies. We therefore follow the ontology model defined in Section 3.2, treating the ontology as a set of axioms. In this section, we define change operations for this ontology model and describe the semantics of change.

Example 8 *We consider a small fragment of our running example – the SWRC ontology – modeling a small research domain, consisting of the following axioms:*
$Employee \sqsubseteq Person$, $Student \sqsubseteq Person$ *(students and employees are persons),*
$Article \sqsubseteq Publication$ *(articles are publications),*
$Article \sqsubseteq \forall author.Person$ *(authors of an article are persons),*
$Article(anArticle)$ *(anArticle is an article),*
$Employee(peter)$, $Employee(paul)$ *(peter and paul are employees),*
$author(anArticle, peter)$, $author(anArticle, paul)$ *(peter and paul are authors of* $anArticle$*).*

Based on the ontology model, we can define ontology change operations.

Definition 14 (Ontology Change Operations) *An* ontology change operation $oco \in OCO$ *is a function* $oco : \mathcal{O} \rightarrow \mathcal{O}$*. Here OCO denotes the set of all change operations.*

An obvious essential question is what we count as a change in an ontology. Do we count every syntactic modification to an ontology, or only those syntactic modifications that affect the semantics of the ontology. A simple example to illustrate the difference is to consider the ontology:

$$C1 \sqsubseteq C2, C1(x), C2(x)$$

Removing the third axiom is clearly a syntactic change, but not a semantic one (the set of models of the ontology does not change, since the removed axiom is also implied by the remaining two). This choice boils down to that of treating an ontology as a set of axioms, or as a set of models. In our work, we have chosen to define an ontology as a set of axioms, allowing us to capture any syntactic modification to an ontology. We consider this approach most suitable as the same logical theory can be encoded by different sets of axioms that have different computational properties (e.g. many ontologies that are formally in OWL Full can be rephrased into an equivalent ontology in OWL DL). The axiom-based approach enables us to distinguish between these two encodings. This choice is in line with other studies of changing ontologies, e.g. [Kle04, Noy05, Sto04a].

For the ontology model of $\mathcal{SHOIN}(\mathbf{D})$, we thus allow the atomic change operations of adding and removing axioms, which we denote with $O\dot{+}\alpha$ and $O\dot{-}\alpha$, respectively. Accordingly, we define the semantics of the change operations: $O\dot{+}\alpha := O \cup \{\alpha\}$ and $O\dot{-}\alpha := O \setminus \{\alpha\}$. Obviously, representing changes at the level of axioms is very fine-grained. However, based on this minimal set of atomic change operations, it is possible to define more complex, higher-level descriptions of ontology changes. Composite ontology change operations can be expressed as a sequence of atomic ontology change operations. The semantics of the sequence is the chaining of the corresponding functions: For some atomic change operations $\text{oco}_1, ..., \text{oco}_n$ we can define $\text{oco}_{composite}(x) = \text{oco}_n \circ ... \circ \text{oco}_1(x) := \text{oco}_n(...(\text{oco}_1))(x)$.

The semantics of change refers to the effect of the ontology change operations and the consistent management of these changes. The consistency of an ontology is defined in terms of consistency conditions, or invariants that must be satisfied by the ontology. We then define rules for maintaining these consistency conditions by generating additional changes.

Definition 15 (Consistency of an Ontology) *We call an ontology O* consistent *with respect to a set of consistency conditions $\mathcal{K}$ iff for all $\kappa \in \mathcal{K}$, O satisfies the consistency condition $\kappa(O)$.*

At this point, we do not make any restriction with respect to the representation of the consistency conditions. They may be expressed for example as logical formulas or functions. In the following, we will further distinguish between structural, logical and user-defined consistency conditions: $\mathcal{K}_S$, $\mathcal{K}_L$, and $\mathcal{K}_U$, respectively. We will call an ontology *structurally consistent*, *logically consistent* and *user-defined consistent*, if the respective consistency conditions are satisfied for the ontology.

Change Generation If we have discovered that an ontology is inconsistent, i.e. some consistency condition is not satisfied, we need to resolve these inconsistencies by generating additional changes that lead to a consistent state. These changes are generated by *resolution functions*:

Definition 16 (Resolution Function) *A* resolution function $\varrho \in \mathcal{P}$ *is a function* $\varrho : \mathcal{O} \times OCO \rightarrow OCO$ *for a consistency condition* κ *that returns for any given ontology* $O in \mathcal{O}$ *and any ontology change operation* $oco \in OCO$ *an additional change operation (which may be composite)* $oco' = \varrho(O, oco)$*, which – applied to the ontology* O *– results in an ontology* $oco'(O)$ *that satisfies the consistency condition* κ*.*

A trivial resolution function would be a function which for a given ontology and change operation simply returns the inverse operation, which effectively means a rejection of the change. Obviously, for a consistent input ontology, applying a change followed by the inverse change will result in a consistent ontology.

In general, there may be many different ways to resolve a particular inconsistency, i.e. different resolution functions may exist. We can imagine a resolution function that initially generates a set of alternative potential change operations, which may be presented to the user who decides for one of the alternatives. Such a resolution function that depends on some external input is compatible with our definition of a resolution function.

We can now define the notion of a resolution strategy that assigns resolution functions to consistency conditions:

Definition 17 (Resolution Strategy) *A* resolution strategy RS *is a total function* $RS : \mathcal{K} \rightarrow \mathcal{P}$*, such that for all* $\kappa \in \mathcal{K}$*,* $\varrho = RS(\kappa)$ *is a resolution function for* κ*.*

The resolution strategy is applied for each ontology change operation in straightforward manner: As long as there are inconsistencies with respect to a consistency condition, we apply the corresponding resolution function.

Please note that a resolution function may generate changes that violate other consistency conditions (resulting in further changes that in turn may violate the previous consistency condition). When defining a resolution strategy, one therefore has to make sure that the application of the resolution strategy terminates, either by prohibiting that a resolution function introduces inconsistencies with respect to any defined consistency condition, or by other means, such as cycle detection.

In the following sections we will introduce various evolution strategies to maintain the structural, logical and user-defined consistency of an ontology.

8.3 Structural Consistency

Structural consistency considers constraints that are defined for the ontology model with respect to the constructs that are allowed to form the elements of the ontology (in our case the axioms). Often this property is also called *validity* with respect to set of constraints. In the context of OWL ontologies, there exist various sublanguages (sometimes also called species), such as OWL DL, OWL Lite, OWL DLP [Vol04]. These sublanguages differ with respect to the constructs that are allowed and can be defined in terms of constraints on the available constructs. The role of these sublanguages is to be able to define ontologies that are "easier to handle", either on a syntactic level to for example allow easier parsing, or on a semantic level to trade some of the expressivity for decreased reasoning complexity. It is thus important that the ontology evolution process provides support for dealing with defined sublanguages: When an ontology evolves, we need to make sure that an ontology does "not leave its sublanguage".

Because of the variety of the sublanguages, it is not possible to operate with a predefined and fixed set of structural consistency conditions. Instead, we allow to define sublanguages in terms of arbitrary structural consistency conditions along with the corresponding resolution functions that ensure that an ontology change operation does not lead out of the defined sublanguage. Please note that because of the definition of the ontology model, we do not allow to construct ontologies outside of the OWL DL language.

8.3.1 Definition of Structural Consistency

In the following we define what it means for an ontology to be structurally consistent with respect to a certain ontology sublanguage. A sublanguage is defined by a set of constraints on the axioms. Typically, these constraints disallow the use of certain constructs or the way these constructs are used.

Some constraints can be defined on a "per-axiom-basis", i.e. they can be validated for the axioms individually. Other constraints restrict the way that axioms are used in combination. In the following we exemplarily show how such consistency conditions can be defined for a particular sublanguage.

Consistency Condition for the OWL Lite sublanguage OWL Lite is a sublanguage of OWL DL that supports only a subset of the OWL language constructs [BvHH+04]. We will now show how it can be defined in terms of a set of structural consistency conditions $\mathcal{K}_S$[1]:

- $\kappa_{S,1}$ disallows the use of disjunction $C \sqcup D$,

[1]Please note that the constraints for the OWL DL language are already directly incorporated into the ontology model itself.

- $\kappa_{S,2}$ disallows the use of negation $\neg C$,
- $\kappa_{S,3}$ restricts the use of the concept $C \sqcap D$ such that C and D be concept names or restrictions,
- $\kappa_{S,4}$ restricts the use of the restriction constructors $\exists R.C$, $\forall R.C$ such that C must be a concept name,
- $\kappa_{S,5}$ limits the values of stated cardinalities to 0 or 1, i.e. $n \in \{0, 1\}$ for all restrictions $\geq n\, R$, $\leq n\, R$,
- $\kappa_{S,6}$ disallows the usage of the oneOf constructor $\{a_1, \ldots, a_n\}$.

8.3.2 Resolving Structural Inconsistencies

Once we have discovered inconsistencies with respect to the defined sublanguage, we have to resolve them. An extreme solution would be to simply remove the axioms that violate the constraints of the sublanguage. This would certainly not meet the expected requirements. A more advanced option is to try to express the invalid axiom(s) in a way that is compatible with the defined sublanguage. In some cases, it may be possible to retain the semantics of the original axioms.

Resolution Strategies for OWL Lite In the following we will present a possible resolution strategy for the OWL Lite sublanguage by defining one resolution function for each of the above consistency conditions in $\mathcal{K}_S$. Although OWL Lite poses many syntactic constraints on the syntax of OWL DL, it is still possible to express complex descriptions using syntactic workarounds, e.g. introducing new concept names and exploiting the implicit negation introduced by disjointness axioms. In fact, using these techniques, OWL Lite can fully capture OWL DL descriptions, except for those containing individual names and cardinality restrictions greater than 1 [HPSvH03].

- $\varrho_{s,1}$ replaces all references to a concept $C \sqcup D$ with references to a new concept name $CorD$, and adds the following axiom: $CorD \equiv \neg(\neg C \sqcap \neg D)$,
- $\varrho_{s,2}$ replaces all references to a concept $\neg C$ in an added axiom with references to a new concept name $NotC$, and adds the following two axioms: $C \equiv \exists R.\top$ and $NotC \equiv \forall R.\bot$, where R is a newly introduced role name,
- $\varrho_{s,3}$ replaces all references to a concept C (or D), where C (or D) is not a concept name or restriction, in concepts $C \sqcap D$ with references to a new concept name aC (or aD), and adds the following axiom: $aC \equiv C$ (or $aD \equiv D$),

- $\varrho_{s,4}$ replaces all references to a concept C (where C is not a concept name) in restrictions $\exists R.C$ or $\forall R.C$ with references to a new concept name aC, and adds the following axiom: $aC \equiv C$.

While these first four resolution functions simply apply syntactic tricks while preserving the semantics, there exist no semantics-preserving resolution functions for the consistency conditions $\kappa_{S,5}$ and $\kappa_{S,6}$.

However, we can either try to approximate the axioms, or in the worst case, simply remove them to ensure structural consistency. We can thus define:

- $\varrho_{s,5}$ replaces all cardinality restrictions $\geq n\, R$ with restrictions $\geq 1\, R$ and removes all axioms containing cardinality restrictions $\leq n\, R$, as these cannot be properly approximated,
- $\varrho_{s,6}$ replaces all occurrences of the concept $\{a_1, \ldots, a_n\}$ with a new concept D and adds assertions $D(a_1), ..., D(a_n)$.

Example 9 *Suppose we want to add to the ontology from Example 8 the axiom* $Publication \sqsubseteq \exists author.\neg Student$*, i.e. stating that all publications must have an author who is not a student. As this axiom violates consistency condition* $\kappa_{S,2}$*, resolution function* $\varrho_{s,2}$ *would generate a composite change that adds the following semantically equivalent axioms instead:* $Publication \sqsubseteq \exists author.NotStudent$*,* $Student \equiv \exists R.\top$*,* $NotStudent \equiv \forall R.\bot$*, resulting in a structurally consistent ontology.*

8.4 Logical Consistency

While the structural consistency is only concerned about whether the ontology conforms to certain structural constraints, the logical consistency addresses the question whether the ontology is "semantically correct", i.e. does not contain contradicting information.

8.4.1 Definition of Logical Consistency

The semantics of the $\mathcal{SHOIN}(\mathbf{D})$ description logic is classically defined via a model-theoretic semantics, which explicates the relationship between the language syntax and the model of a domain: An interpretation $\mathcal{I} = (\triangle^I, \cdot^I)$ consists of a domain set $\triangle^I$ and an interpretation function $\cdot^I$, which maps from individuals, concepts and roles to elements of the domain, subsets of the domain and binary relations on the domain, respectively. An interpretation $\mathcal{I}$ satisfies an ontology O, if it satisfies each axiom in O. Axioms thus result in semantic conditions on the interpretations. Consequently, contradicting axioms will allow no possible interpretation.

We can thus define a consistency condition for *logical consistency* κ_L that is satisfied for an ontology O if O is satisfiable, i.e. if O has a model.

Please note that because of the monotonicity of the considered logic, an ontology can only become logically inconsistent by adding axioms: If a set of axioms is satisfiable, it will still be satisfiable when any axiom is deleted. Therefore, we only need to check the consistency for ontology change operations that add axioms to the ontology.

Effectively, if $O \cup \{\alpha\}$ is inconsistent, in order to keep the resulting ontology consistent some of the axioms in the ontology O have to be removed. In this sense, the add-operation and the remove-operation are similar to the belief revision operation and the belief contraction operation in the theories of belief revision [FPA05]. However, note that the main difference between our approach here and the belief revision approach is that the belief revision is a model-based approach, where a change operation is defined in terms of logical consequences, whereas our approach is axiom-based (c.f. discussion following Definition 14).

Example 10 *Suppose, we start out with the initial ontology from Example 8 extended with the axiom* $Student \sqsubseteq \neg Employee$ *(Students and Employees are disjoint). This ontology is logically consistent.*

Suppose we now want to add the axiom $Student(peter)$*, stating that the individual* $peter$ *is a student. Obviously, this ontology change operation would result in an inconsistent ontology, as we have stated that students and employees are disjoint on the one hand, and that* $peter$ *is a student and a employee on the other hand.*

Now, there may be many ways how to resolve this inconsistency. One possibility would be to reject the change $Student(peter)$*. Alternatively, we could also remove the assertion* $Employee(peter)$*. However, if both of these assertions are correct, the user may not be happy with either decision. The most intuitive one may be to retract the axiom* $Student \sqsubseteq \neg Employee$*, but also this may not satisfy the user. A further, more complex change, would be to introduce a new concept* $PhdStudent$*, which need not be disjoint with employees.*

8.4.2 Resolving Logical Inconsistencies

In the following, we present resolution functions that allow us to define resolution strategies to ensure logical consistency. The goal of these resolution functions is to determine a set of axioms to remove in order to obtain a logically consistent ontology with "minimal impact" on the existing ontology. Obviously, the definition of minimal impact may depend on the particular user requirements. A very simple definition could be that the number of axioms to be removed should be minimized. More advanced definitions could include a notion of confidence or relevance of the axioms. Based on this notion of "minimal impact" we can define an algorithm that generates a minimal number of changes that result in a maximal consistent subontology.

However, in many cases it will not be feasible to resolve logical inconsistencies in a fully automated manner. We therefore also present a second, alternative approach

for resolving inconsistencies that allows the interaction of the user to determine which changes should be generated. Unlike the first approach, this approach tries to localize the inconsistencies by determining a minimal inconsistent subontology.

Alternative 1: Finding a consistent subontology

In our model we assume that the ontology change operations should lead from one consistent ontology to another consistent ontology. If an ontology change operation (adding an axiom, $O \dot{+} \alpha$) would lead to an inconsistent ontology, we need to resolve the inconsistency by finding an appropriate subontology $O' \subset O$ (with $\alpha \in O'$) that is consistent. We do this by finding a maximal consistent subontology:

Definition 18 (Maximal consistent subontology) *An ontology O' is a* maximal consistent subontology *of O, if $O' \subset O$ and O' is logically consistent and every O'' with $O' \subset O'' \subseteq O$ is logically inconsistent.*

Intuitively, this definition states that no axiom from O can be added to O' without losing consistency. In general, there may be many maximal consistent subontologies O'. It is up to the resolution strategy and the user to determine the appropriate subontology to be chosen.

The main idea is that we start out with the inconsistent ontology $O \cup \{\alpha\}$ and iteratively remove axioms until we obtain a consistent ontology. Here, it is important how we determine which axioms should be removed. This can be realized using a *selection function*. The quality of the selection function is critical for two reasons: First, as we potentially have to search all possible subsets of axioms in O for the maximal consistent ontology, we need to prune the search space by trying to find the *relevant* axioms that cause the inconsistency. Second, we need to make sure that we remove the *dispensable* axioms. (Please note that a more advanced strategy could consider to only remove parts of the axiom.)

The first problem of finding the axioms that cause the inconsistency can be targeted by considering that there must be some "connection" between these problematic axioms. We formalize this notion using a *connectedness relation* between pairs of axioms.

Definition 19 (Connection relation) *A connection relation $\mathcal{C}$ is a set of axiom pairs, namely, $\mathcal{C} \subseteq \mathcal{O} \times \mathcal{O}$.*

A very simple, but useful connection is that of the direct structural connection relation, as first proposed in [CPW01]:

Definition 20 (Direct Structural Connection) *Two axioms α and β are* directly structurally connected *– denoted with connected(α, β) –, if there exists an ontology entity $e \in N_C \cup N_{I_a} \cup N_{I_c} \cup N_{R_a} \cup N_{R_c}$ that occurs in both α and β.*

The second problem of only removing dispensable axioms requires more semantic selection functions. These semantic selection functions can for example exploit information about the confidence in the axioms that allows us to remove less probable axioms. Such information is for example available in probabilistic ontology models, such as [DP04].

In the following, we present an algorithm (c.f. Algorithm 2) for finding (at least) one maximal consistent subontology using the definition of structural connectedness (c.f. Definition 20):

Algorithm 2 Determine consistent subontology for adding axiom α to ontology O

```
1:  Ω := {O ∪ {α}}
2:  repeat
3:    Ω' := ∅
4:    for all O' ∈ Ω do
5:      for all β1 ∈ O' \ {α} do
6:        if there is a β2 ∈ ({α} ∪ (O \ O')) such that connected(β1, β2) then
7:          Ω' := Ω' ∪ {O' \ {β1}}
8:        end if
9:      end for
10:   end for
11:   Ω := Ω'
12: until there exists an O' ∈ Ω such that O' is consistent
```

We maintain a set of possible candidate subontologies Ω, which initially contains only $O \cup \{\alpha\}$ (c.f. line 1), i.e. the consistent ontology O before the change and the added axiom α. In every iteration, we generate a new set of candidate ontologies (line 3) by removing one axiom β_1 from each candidate ontology (line 7) that is structurally connected with α or an already removed axiom (in $O \setminus O'$, line 6), until at least one of the candidate ontologies is a consistent subontology (line 12).

The properties of the algorithm (efficiency, completeness) will depend on the properties of the connectedness relation. The above definition of structural connectedness provides good heuristics to efficiently find a maximal consistent subontology, but is not complete for the case where axioms causing an inconsistency are not structurally connected at all.

Example 11 *We now show how Algorithm 2 can be used to maintain consistency. Consider a change operation oco_1 that adds the axiom $\alpha = PhDStudent \sqsubseteq Employee$. Then $oco_1(O_1)$ results in an inconsistent ontology.*

Algorithm 2 starts with $O_1 \dot{+} \alpha$ as element of the set of potential ontologies. In the first iteration, a set of new potential ontologies is created by removing one of the axioms

that are structurally connected with the α. These axioms are: $PhDStudent(peter)$, $Employee \sqsubseteq \neg Student$, $PhDStudent \sqsubseteq Student$ and $Employee \sqsubseteq Person$.

The removal of either $PhDStudent(peter)$, $PhDStudent \sqsubseteq Student$ or $Employee \sqsubseteq \neg Student$ will result in a maximal consistent subontology. For the decision which axiom should be removed from the ontology, one can rely on further background information indicating the relevance of the axioms, or on interaction with the user. For the following examples, we assume that the resulting ontology O_2 is created by removing the axiom $Student \sqsubseteq \neg Employee$, i.e. $O_2 = O_1 \dot{+} PhDStudent \sqsubseteq Employee \dot{-} Student \sqsubseteq \neg Employee$.

Alternative 2: Localizing the inconsistency

In the second alternative, we do not try to find a consistent subontology, instead we try to find a minimal inconsistent subontology, i.e. a minimal set of contradicting axiom. We call this process *Localizing the inconsistency*. Once we have localized this minimal set, we present it to the user. Typically, this set is considerably smaller than the entire ontology, such that it will be easier for the user to decide how to resolve the inconsistency.

Definition 21 (Minimal inconsistent subontology) *An ontology O' is a* minimal inconsistent subontology *of O, if $O' \subseteq O$ and O' is inconsistent and for all O'' with $O'' \subset O'$, O'' is consistent.*

Intuitively, this definition states that the removal of any axiom from O' will result in a consistent ontology.

Again using the definition of connectedness, we can realize an algorithm (c.f. Algorithm 3) to find a minimal inconsistent ontology: We maintain a set Ω with candidate ontologies, which initially only consists of the added axiom $\{\alpha\}$ (c.f. line 1). As long as we have not found an inconsistent subontology, we add one structurally connected axiom (line 6) to each candidate ontology (line 7).

Because of the minimality of the obtained inconsistent ontology, it is sufficient to remove any of the axioms to resolve the inconsistency. The minimal inconsistent ontology can be presented to the user, who can select the appropriate axiom to remove. It may be possible that one added axiom introduced multiple inconsistencies. For this case, the above algorithm has to be applied iteratively.

Example 12 *We now show how Algorithm 3 can be used to localize the inconsistency in our running example, which has been introduced by adding the axiom Student(peter). Applying the algorithm, we start out with the candidate ontology $\Omega := \{\{Student(peter)\}\}$. Adding the structurally connected axioms, we obtain: $\Omega := \{\{Student(peter), Employee(peter)\}, \{Student(peter), Student \sqsubseteq Person\}, \{Student(peter), Student \sqsubseteq \neg Employee\}, \{Student(peter), Employee(ljiljana)\}, \{Student(peter), author(anArticle, peter)\}\}$.*

Algorithm 3 Localize inconsistency introduced by adding axiom α to ontology O

```
1:  Ω := {{α}}
2:  repeat
3:    Ω' := ∅
4:    for all O' ∈ Ω do
5:      for all β1 ∈ O \ O' do
6:        if there is a β2 ∈ O' such that connected(β1, β2) then
7:          Ω' := Ω' ∪ {O' ∪ {β1}}
8:        end if
9:      end for
10:   end for
11:   Ω := Ω'
12: until there exists an O' ∈ Ω such that O' is inconsistent
```

All of these candidate ontologies are still consistent. In the next iteration, adding the structurally connected axiom $Student \sqsubseteq \neg Employee$ *to the candidate ontology* $\{Student(peter), Employee(peter)\}$ *will result in the minimal inconsistent subontology* $\{Student(peter), Employee(peter), Student \sqsubseteq \neg Employee\}$.

The removal of any of these axioms (which one is to be decided by the user), will lead to a consistent ontology.

Using contextual information for the resolution of changes. Instead of leaving the final choice of how to resolve the inconsistency to the user, we can alternatively rely on contextual information, e.g. about the certainty or importance of axioms, to automate the selection of axioms. For example, in [HV05] we show how the algorithm can be applied for the evolution task of incremental ontology learning. We here apply ontology learning algorithms to generate ontologies based on a Learned Ontology Model. The ontology learning algorithms are largely based on heuristics and statistics; it is thus inherent in the ontology learning process that the acquired ontologies represent uncertain and possibly contradicting knowledge. In the Learned Ontology Model, we represent uncertainty as contextual annotations capturing the confidence about the correctness of the ontology elements. In the learning process, as new ontology elements are learned and added to the ontology, we apply the methods of consistent ontology evolution, in particular we apply Alternative 2 to localize inconsistencies as they are introduced. In the resolution of the changes we remove the axioms that have the lowest confidence, i.e. those axioms that are most likely incorrect. We are thus able to incrementally evolve an ontology that is (1) consistent and (2) captures the information with the highest confidence. For details of this process and evaluation results, we refer the reader to [HV05].

8.5 User-defined Consistency

The user-defined consistency takes into account particular user requirements that need to be expressed "outside" of the ontology language itself. While an ontology may be structurally consistent (e.g. be a syntactically correct ontology according to a particular OWL sublanguage) and may be logically consistent, it may still violate some user requirements. With the term *user-defined consistency conditions* we here subsume a number of different properties of an ontology that are sometimes more specifically called constraints or validity conditions. We can identify two types of user-defined consistency conditions: *generic* and *domain dependent*.

Generic consistency conditions are applicable across domains and represent e.g. best design practice or modeling quality criteria. For example, OntoClean [WG01] formalizes a set of meta-properties representing the philosophical notions of *rigidity*, *identity*, *unity*, and *dependence*. These meta-properties are assigned to properties (corresponding to concepts in DL terminology) of the ontology. Constraints on the taxonomic relationships define the consistency of the ontology, e.g. a non-rigid property cannot subsume a rigid property.

Domain dependent consistency conditions take into account the semantics of a particular formalism of the domain. An example are consistency conditions for the OWL-S process model [SASS04] to verify web service descriptions.

In the following we exemplarily show how user-defined consistency conditions and corresponding resolution functions can be described to ensure *modeling quality conditions*. Such modeling quality conditions cover redundancy, misplaced properties, missing properties, etc. We refer the reader to [Sto04a] for a complete reference.

One example of redundancy is *concept hierarchy redundancy*. If a direct superconcept of a concept can be reached through a non-direct path, then the direct link is redundant. We can thus define a consistency condition that disallows concept hierarchy redundancy: $\kappa_{U,1}$ is satisfied if for all axioms $C_1 \sqsubseteq C_n$ in O there exist no axioms in O with $C_1 \sqsubseteq C_2$, ..., $C_{n-1} \sqsubseteq C_n$. We can further define a corresponding resolution function $\varrho_{U,1}$ that ensures this consistency condition by generating a change operation that removes the redundant axiom $C_1 \sqsubseteq C_n$.

Example 13 *Suppose, we start out with the ontology from our Example 8, i.e. the initial example extended with the axiom $Professor \sqsubseteq Person$ (a professor is a person). This ontology is consistent with respect to the consistency definition $\kappa_{U,1}$.*

Suppose we now want to add the axiom $Professor \sqsubseteq Employee$, stating that the professor is an employee. Obviously, this ontology change operation would result in an ontology that is inconsistent with respect to $\kappa_{U,1}$ since there is an alternative path (through the concept $Employee$) between the concept $Professor$ and its direct superconcept $Person$. The resolution function $\varrho_{U,1}$ would generate a change operation that removes the axiom $Professor \sqsubseteq Person$.

8.6 Alternatives in Dealing with Inconsistencies

The approach of consistent ontology evolution presented in the previous sections addresses the use case, where changing an initially consistent ontology potentially introduces inconsistencies. This typically occurs in settings where one is in control of changes and needs support for maintaining consistency during evolution.

However, we can identify other use cases that require dealing with inconsistencies in changing ontologies. For example, re-using ontologies in open settings such as the Web might include the retrieval of inconsistent ontologies that should be fixed before usage. In some cases consistency cannot be guaranteed at all and inconsistencies cannot be repaired, still one wants to derive meaningful answers when reasoning. Often this is the case when the schema-level and the instance-level of an ontology are evolved separately without synchronizing the changes continuously. Finally, when applying an ontology one faces the challenge to decide whether the usage of other, e.g. newer, versions of this ontology might lead to inconsistencies in an application, or, in other words, whether the versions are compatible with respect to certain aspects. While the former use cases typically occur during the development of ontologies, the latter ones illustrate the handling of inconsistencies during the runtime of ontology-based applications.

In this section we compare different approaches to address these use cases within a framework for combining currently separate methods for inconsistency-handling in changing ontologies. This framework was first presented in [HvHH+05] and builds on common definitions of ontology change and consistency. In particular, we focus on logical consistency. To meet the requirements of the above mentioned use cases our framework consists of the following main components:

Consistent Ontology Evolution is the process of managing ontology changes by preserving the consistency of the ontology with respect to a given notion of consistency. The consistency of an ontology is defined in terms of consistency conditions, or invariants that must be satisfied by the ontology.

Repairing Inconsistencies involves a process of diagnosis and repair: first the cause (or: a set of potential causes) of the inconsistency needs to be determined, which can subsequently be repaired.

Reasoning with Inconsistent Ontologies does not try to avoid or repair the inconsistency (as in the previous two approaches), but simply tries to "live with it" by trying to return meaningful answers to queries, even though the ontology is inconsistent.

Ontology Versioning manages the relations between different versions of an ontology, and a notion of compatibility with such versions. One such compatibility relation is inconsistency: even though two versions of an ontology may each be consistent in themselves, they might derive some opposite conclusions, and would then be mutually inconsistent.

Consistent ontology evolution has already been described in detail in the previous sections. In the following, we describe the remaining three alternative strategies in terms of the notions introduced in the previous section and provide a comparison.

8.6.1 Repairing Inconsistencies

The most straightforward approach to inconsistencies is to repair them when they are detected [SC03]. Repairing an inconsistency actually consists of two tasks: Locating Inconsistencies and Resolving Inconsistency. The task of repairing inconsistencies can thus be defined as: For an inconsistent ontology O we generate a change operation oco such that $O' = oco(O)$ results in a consistent ontology O'.

Locating Inconsistencies As a first step, the source of the inconsistency has to be detected. Normally, the source is a set of axioms that when being part of the model at the same time make it inconsistent.

An algorithm to find a subontology which leads to an unsatisfiable concept (adopted from [SC03]) can use similar ideas like those for consistent ontology evolution. The main difference is that the latter assumes that the intended minimal inconsistent ontology would contain an added axiom α, whereas the former has no such requirement but starting with an unsatisfiable concept C for the connection checking[2]. Algorithm 4 uses the increment-reduction strategy to find a minimal subontology for an unsatisfiable concept. Namely, the algorithm finds a subset of the ontology in which the concept is unsatisfiable first (lines 2 to 8), then reduces the redundant axioms from the subset (lines 9-13).

Based on those detected subsets for all unsatisfiable concepts, we can find minimal subsets of the ontology O which leads to all unsatisfiable concepts[SC03]. That can be used by an ontology engineer to repair the ontology in order to avoid all unsatisfiable concepts.

Resolving Inconsistency Once the source of an inconsistency has been found, the conflict between the identified set of axioms has be to resolved. This task again is difficult, because in most cases there is no unique way of resolving a conflict but a set of alternatives. Often, there are no logical criteria selecting the best resolution. A common approach is to let the user resolve the conflict after it has been located.

Example 14 *We again use the running example introduced in Example 11. Assume that we start out with the inconsistent ontology* $O_3 = \{Employee \sqsubseteq Person,\ Student \sqsubseteq$

[2]In order to do so, we extend the direct structural connection relation on concept sets, so that we can say something like an axiom β is connected with a concept C, i.e., $connected(\beta, C)$. It is easy to see that it does not change the definition.

Algorithm 4 Localize a minimal subset of O in which a concept C is unsatisfiable

```
1: Ω := ∅
2: repeat
3:    for all β1 ∈ O \ Ω do
4:       if there is a β2 ∈ Ω such that connected(β1, β2) or connected(β1, C) then
5:          Ω := Ω ∪ {β1}
6:       end if
7:    end for
8: until C is unsatisfiable in Ω
9: for all β ∈ Ω do
10:   if C is unsatisfiable in Ω − {β} then
11:      Ω := Ω − {β}
12:   end if
13: end for
```

$Person$, $PhDStudent \sqsubseteq Student$, $Employee \sqsubseteq \neg Student$, $PhDStudent \sqsubseteq Employee$, $PhDStudent(peter)\}$.

In this example the concept $PhDStudent$ is unsatisfiable. Starting with this unsatisfiable concept the algorithm finds the connected set $O_{31} = \{PhDStudent \sqsubseteq Student$, $PhDStudent \sqsubseteq Employee$, $PhDStudent(peter)\}$. The concept $PhDStudent$ is still satisfiable in O_{31}. Extending O_{31} with the connection relation the algorithm gets O_3. Reducing the redundant axioms, the algorithm finds the set $O_{32} = \{PhDStudent \sqsubseteq Student$, $Employee \sqsubseteq \neg Student$, $PhDStudent \sqsubseteq Employee\}$. Since $PhdStudent$ is the only unsatisfiable concept in this example, the ontology engineer can focus on the set O_{32} to repair O_3.

There is a relatively well studied method for diagnosis, with a straightforward definitions: diagnosis is the smallest set of axioms that need to be removed to make the ontology consistent. These diagnoses can be calculated relatively easily on the basis of the minimal inconsistent subontologies. So, this covers the two parts of localizing and repairing inconsistencies (repairing an incoherent model by removing the minimal diagnoses).

8.6.2 Reasoning with Inconsistent Ontologies

In some cases it is unavoidable to live with inconsistencies, if consistency cannot be guaranteed and inconsistencies cannot be repaired. Nevertheless, there is still a need to reason about ontologies in order to support information access and integration of new information. We can summarize the task of reasoning with inconsistent ontologies: For a possibly inconsistent ontology O and a boolean query, the task of inconsistency reasoning is to return a meaningful query answer. A query answer to a boolean query

$O \mid\!\approx \alpha?$ is obtained by the evaluation of the entailment relation $\mid\!\approx$, which classically may be either *yes* ($O \mid\!\approx \alpha$), or *no* ($O \not\mid\!\approx \alpha$).

Definition 22 (Consistent Query Answer) *For an ontology O and an entailment relation $\mid\!\approx$, an answer '$O \mid\!\approx \alpha$' is said to be consistent if $O \not\mid\!\approx \neg\alpha$.*

For a consistent ontology O, its query answer is always consistent under a standard entailment. Namely, the consequence set $\{\alpha : O \models \alpha\}$ is consistent.

As shown above, the standard entailment is explosive: Any formula is a logical consequence of an inconsistent ontology. Therefore, conclusions drawn from an inconsistent ontology by classical inference may be completely meaningless. For an inconsistency reasoner it is expected that it is able to return meaningful answers to queries, given an inconsistent ontology. In the case of a consistent ontology O, classical reasoning is sound, i.e., a formula ϕ deduced from O holds in every model of O. This definition is not preferable for an inconsistent ontology O as every formula is a consequence of O using a standard entailment $\models$. However, often only a small part of O has been incorrectly constructed or modeled, while the remainder of O is correct. Therefore, in [HvHtT05] the following definition of meaningfulness has been proposed:

Definition 23 (Meaningfulness) *A query answer to a query $O \mid\!\approx \alpha?$ is meaningful iff the following two conditions are satisfied:*

1. soundness: *the answer is a consequence of a consistent subontology of O under the standard entailment $\models$,*

2. consistency: *the answer is a consistent query answer under the entailment $\mid\!\approx$.*

The general strategy for processing inconsistent ontologies is: given a connection/relevance relation (c.f. Definition 19), we select some consistent subontology from an inconsistent ontology. Then we apply standard reasoning on the selected subontology to find meaningful answers. If a meaningful answer cannot be found, the relevance degree of the selection function is made less restrictive thereby extending the consistent subontology for further reasoning. If an inconsistent subset is selected, we call the *over-determined processing* (ODP)[HvHtT05]. One of the ODP strategies is to find the set of the maximal consistent subontologies of the selected set. A linear extension strategy with an ODP for the evaluation of a query '$O \mid\!\approx \alpha?$' is described in Algorithm 5: We start with an empty ontology Ω in line 1. We incrementally add structurally connected axioms (line 3) until we have identified a consistent subontology from which we can derive the query answer (line 15). The consistency of the subontology is ensured with the over-determined processing in lines 8 to 14. In [HvHtT05] it is proven that the answers which are obtained by the linear extension strategy are meaningful.

Algorithm 5 Linear extension strategy for the evaluation of query $O \not\approx \alpha$?

1: $\Omega := \emptyset$
2: **repeat**
3: $\Omega' := \{\beta_1 \in O \setminus \Omega$: there exists a $\beta_2 \in \Omega \cup \{\alpha\}$ such that $connected(\beta_1, \beta_2)\}$
4: **if** $\Omega' = \emptyset$ **then**
5: **return** $O \not\approx \alpha$
6: **end if**
7: $\Omega := \Omega \cup \Omega'$
8: **if** Ω inconsistent **then**
9: $\Omega'' := maximal\_consistent\_subontology(\Omega)$
10: **if** $\Omega'' \models \alpha$ **then**
11: **return** $O \not\approx \alpha$
12: **else return** $O \not\approx \alpha$
13: **end if**
14: **end if**
15: **until** $\Omega \models \alpha$
16: **return** $O \not\approx \alpha$

Example 15 *Consider the inconsistent ontology* $O_3 = \{Employee \sqsubseteq Person$, $Student \sqsubseteq Person$, $PhDStudent \sqsubseteq Student$, $PhDStudent \sqsubseteq Employee$, $Employee \sqsubseteq \neg Student$, $PhDStudent(peter)\}$

Assume we want to ask the query $O_3 \not\approx Student(peter)$?. *Using standard entailment we would obtain no meaningful answer, as both* $Student(peter)$ *and* $\neg Student(peter)$ *are entailed by the ontology. By the linear extension on the connection relation with* $Student(peter)$*, the algorithm will construct the ontology* $\Omega = \{PhDStudent(peter), PhDStudent \sqsubseteq Employee, PhDStudent \sqsubseteq Student\}$. *This ontology* Ω *is consistent, and* $\Omega \models \alpha$. *Thus, the algorithm concludes that* $O_3 \not\approx Student(peter)$.

8.6.3 Multi-Version Reasoning

Multi-version reasoning is an approach that tries to cope with possible inconsistencies in changing ontologies by considering not only the latest version of an ontology, but all previous versions as well. We consider the sequence of ontologies $O_1 \prec \cdots \prec O_n$ where the ordering relation is defined as:

$$O_i \prec O_j \Leftrightarrow \exists \text{oco}_{composite} : \text{oco}_{composite}(O_i) = O_j$$

Intuitively, O_n is the current version of the ontology. $O_1, \cdots, O_{n-1}$ are older versions of the same ontology that have been created from the respective previous ontology

in terms of a composite change action. We can assume that each of the ontologies is consistent. Further, we assume that an application expresses its requirements for compatibility as an expectation α, for which there is an ontology O_i in the sequence such that $O_i \cup \{\alpha\}$ is consistent. Expectations can thus be thought of as statements that are expected to hold in an ontology.

Based on these assumptions, the task of ensuring consistency reduces to the task of finding the right version O_i of the ontology in the sequence of versions. This task requires the ability to determine the satisfiability of certain expressions across the different versions of the ontology. This can be done using an extension of the ontology language called $\mathcal{L}+$ with the operator $\mathbf{PreviousVersion}\phi$, which is read as '$\phi$ holds in the previous version', the operator $\mathbf{AllPriorVersions}\phi$, which is read as '$\phi$ holds in all prior versions', and the operator $\mathbf{SomePriorVersion}\phi$, which is read as '$\phi$ holds in some prior versions'.

Using these basic operators, we can define a rich set of query operators for asking specific questions about specific versions of the ontology and relations between them. In the case where $O_n \cup \{\alpha\}$ is inconsistent, we can for example check whether the previous version can be used ($\mathbf{PreviousVersion}\,\alpha$) and whether there is a version at all that can be used instead ($\mathbf{SomePriorVersion}\,\alpha$). For the formal semantics of these operators we refer the reader to [HS05c].

Example 16 *Consider we have an ordered relation of ontologies $O_1 \prec O_2$, using the ontologies from Example 11. Now assume a compatibility criteria that has to be fulfilled for compatibility: $\alpha = Employee(peter)$, i.e. a knowledge base in which Peter is an employee. The latest version O_2 is compatible with the compatibility criteria α as $O_2 \cup \{\alpha\}$ is consistent. However, O_1 does not meet the compatibility requirements, as $O_1 \cup \{\alpha\}$ is inconsistent (It still contained the axiom stating the disjointness of students and employees). In fact, it holds that $\mathbf{AllPriorVersions}\,\neg Employee(peter)$.*

8.6.4 Comparison and Evaluation

We are going to compare the four approaches dealing with inconsistency, and make an evaluation on them. By the evaluation, we want to suggest several guidelines for system developers to know under which circumstance which approach is more appropriate.

A first major difference that is revealed by the formal analysis in the previous section is the fact that the different methods for dealing with inconsistent ontologies actually have very different functionality (their input/output-relations are rather different). Consequently, they solve rather different tasks, and are suited for different use-cases. The situation is summarized in Table 8.1.

Dependence on query. First, this table shows that two of the methods depend on which user-query is given to the ontology (reasoning with inconsistency and multi-

Approach	Input	Output
Consistent Evolution	Consistent Ontology, Change	Consistent Ontology
Inconsistency Repair	Inconsistent Ontology	Consistent Ontology
Inconsistency Reasoning	Inconsistent Ontology, Query	Meaningful Answer
Multi-version Reasoning	Versions of Ontologies, Query	Consistent Answer

Table 8.1: Comparison of Approaches to Dealing with Inconsistencies

version reasoning). Consequently, these two methods are only applicable at *runtime*, when a user interacts with the ontology. The other two methods (ontology evolution and inconsistency repair) are independent of user-queries, and can thus already be applied at ontology *development time*.

Known or unknown change. The two methods that are applicable at ontology development time are actually very similar (as is apparent from sections 8.4 and 8.6.1). A crucial difference is that the first of these (ontology evolution) requires knowledge of the change that caused the ontology to become inconsistent: algorithm 2 requires the change α to be known, which is not the case with 4. This is clearly a restriction on the applicability of ontology evolution, which comes in exchange for the benefit of a simpler algorithm.

Known or unknown history. The two query-dependent approaches also differ in their respective input-requirements: multi-version reasoning requires a *history* of ontology-versions to be available, which is a very strong demand, often not feasible in many settings, in particular in combination with its runtime usage.

Heuristics. Another difference between the various approaches is the extent to which they employ heuristics: in reasoning with inconsistency, one heuristically chooses a consistent subontology that is good enough to answer the query (it need not be minimal, just small enough to be consistent, and large enough to answer the query). In contrast, both Evolution and Repair aim at the *smallest* impact on the inconsistent ontology.

Efficiency. Finally, one would expect the various approaches to differ drastically in their computational efficiency. Some observations can be made immediately: the Evolutionary approach exploits the knowledge about the cause of the inconsistency, and can therefore be more efficient than Repair, which does not have access to this information. However, the cost of all of the algorithms described here are dominated by untractable operations such as checking the unsatisfiability of a concept or the inconsistency of an entire ontology. Consequently, worst-case complexity analysis is not going to tell us

anything interesting here. Instead, work will have to be done on average-case complexity analysis and experiments with realistic datasets to gain more insight into the relative costs of each of the approaches.

Knowledge Requirements. Finally, the approaches differ in the knowledge that is required to operate them:

- the repair approach requires the ontology developers to have sufficient domain knowledge to decide which part of the ontology should be removed to recover consistency. On the other hand, once done, it needs no additional expertise from the ontology users.
- Reasoning with inconsistencies on the other hand imposes no knowledge requirements on the developers, but requires some (weak) knowledge from the users to determine whether a query answer is acceptable.
- Ontology versioning places again a heavy knowledge requirement on the user in order to decide which version is most suitable for their application.

8.7 Related Work

In the last decade there has been very active research in the area of ontology engineering. The majority of research studies in this area are focused on construction issues. However, coping with the changes and providing maintenance facilities require a dedicated approach for ontology evolution. The evolution of ontologies has been addressed by different researchers by defining change operations and change representations for ontology languages.

Change operations have been proposed for specific ontology languages. In particular change operations have been defined for OKBC, OWL [Kle04] and for the KAON ontology language [Sto04a]. All approaches distinguish between atomic and complex changes. Different ways of representing ontological changes have been proposed: besides the obvious representation as a change log that contains a sequence of operations, authors have proposed to represent changes in terms of mappings between two versions of an ontology [NM03]. [Sto04a] defines an ontology evolution process which we have adapted for our work. However, the semantics of change in [Sto04a] focuses on the KAON ontology model, which is fundamentally different from the OWL ontology model, as described earlier. A taxonomy of ontology changes for the OWL ontologies can be found in [Kle04]. However, in [Kle04] the ontology model follows a more object-oriented view, whereas we follow the axiomatic ontology model of [PSHH04].

While there exist significant differences between schema evolution and ontology evolution, as elaborated in [NK02], particular aspects of schema evolution in databases are

relevant for our work. [Rod95] provides an excellent survey of the main issues concerned. A sound and complete axiomatic model for dynamic schema evolution in object-based systems is described in [PÖ97]. This is the first effort in developing a formal basis for the schema evolution research. The authors define consistency of a schema with a fixed set of invariants or consistency conditions that are tailored to the data model.

However, in the context of OWL ontologies, the notion of consistency is much more multifaceted. First, the existing work only considers structural consistency. Not only is the set of structural constraints different due to the difference in the underlying models. We further support the evolution of various fragments (sublanguages) of OWL that are defined using different structural constraints. Furthermore, we consider the notions of logical and user-defined consistency.

The problem of preserving integrity in the case of changes is also present for ontology evolution. On the one hand the problem is harder here as ontologies are often encoded using a logical language where changes can quickly lead to logical inconsistency that cannot directly be determined by looking at the change operation. On the other hand, there are logical reasoners that can be used to detect inconsistencies both within the ontology and with respect to instance data. As this kind of reasoning is often costly, heuristic approaches for determining inconsistencies have been proposed [Kle04, SK03]. While deciding whether an ontology is consistent or not can easily be done using existing technologies, repairing inconsistencies in ontologies is a difficult problem. However, the problem of diagnosing inconsistent ontologies has received increased attention, which is a prerequisite for a successful repair. For example, Schlobach and colleagues [SC03, Sch05] propose a diagnosis based on identifying MIPS (Minimal Incoherence Preserving Sub-TBoxes) and MUPS (Minimal Unsatisfiability Preserving Sub-TBoxes), which are used to generate explanations for unsatisfiable concepts.

[PT06] presents an approach to locating and resolving inconsistencies that in principle follows our idea of consistent ontology evolution, but it does so by extending existing tableaux algorithms to determine the axioms causing an inconsistency. The advantage is that the tableaux can be exploited in the resolution of the inconsistency. The disadvantage obviously is that it is bound to a specific reasoner and requires access to the internal data structures of that reasoner. It is therefore referred to as a "glass-box" approach. A similar glass-box approach is also taken by the Pellet reasoner within the SWOOP ontology editor [PSK05, KPSCG06], which offers a debugging mode in which explanations of inconsistencies are provided as a result of the reasoning process. In contrast, our work follows a black-box approach that requires no insights into the reasoning process.

Regarding the notion of logical consistency, the research done in belief revision is of interest: Here, the revision problem is concerned about resolving contradictions by minimal mutilation of a set of beliefs. The combination of classical approaches with description logics is subject of ongoing research: In [FPA05, FPA06] the authors have shown that the AGM postulates [AGM85], which are the foundations of most work in belief revision, cannot be directly applied, as their underlying assumptions generally fail

for DLs. In [QLB06] the authors generalize the AGM postulates to DLs and propose specific revision operators.

Finally, there are several tools that support species validation (corresponding to our structural consistency) or localizing inconsistencies in ontologies. For example, the OWL Protege Plugin [KFNM04] provides species validation including explanations where certain problems occurred. The OWL Protege Plugin also provides explanations on ontology changes, i.e. new subsumptions that have been inferred, logical inconsistencies that have been introduced (based on RACER reasoning services). [BMK+04] presents a "symptom" ontology describing inconsistencies and errors in ontologies. It provides a classification of inconsistencies according to various levels of severity. However, there is no support for preserving consistency in the case that consistency conditions are violated in the presence of ontology changes.

8.8 Conclusions

Knowledge intensive applications in distributed information systems will not be able to ignore the issue of inconsistent knowledge in general, and of inconsistent ontologies in particular. In this chapter we have presented an approach to formalize the semantics of change for the OWL ontology language (for OWL DL and sublanguages in particular), embedded in a generic process for ontology evolution. Our formalization of the semantics of change allows to define various consistency conditions – grouped in structural, logical, and user-defined consistency – and to define resolution strategies that assign resolution functions to that ensure these consistency conditions are satisfied as the ontology evolves. We have shown exemplarily, how such resolution strategies can be realized for various purposes. Finally, we have compared our approach of consistent ontology evolution with other approaches to deal with inconsistencies in evolving ontologies.

Chapter 9

Collaborative Evolution of Personal Ontologies

In the previous chapter we have addressed the semantics of change to guarantee the consistent evolution of ontologies as an important phase of the ontology evolution process. In this chapter we turn to another important aspect of evolution in distributed information systems: the existence of multiple, distributed and frequently changing views on the domain. Existing approaches of ontology-based information access almost always assume a setting where information providers share an ontology that is used to access the information. In a decentralized setting, this assumption does no longer hold. We rather face the situation, where individual nodes maintain their own view of the domain in the form of personal ontologies.

A typical application scenario is an information system for a community of users that relies on ontologies to structure its contents and facilitate browsing and searching (e.g., ACM Topic Hierarchy for computer science literature, Amazon product taxonomy, etc.). As in heterogenous communities users typically will use different parts of such ontologies with varying intensity, customization and personalization of the ontologies is desirable. The sheer size of e.g. the ACM Topic Hierarchy makes it quite difficult for users to easily locate topics which are relevant for them. Here, personal ontologies allow to reflect the interests of users at certain times. Interests might change as well as the available data, therefore the personalization requires quite naturally support for the evolution of personal ontologies.

Obviously, the dimension of dealing with multiple ontologies poses new challenges for the evolution, but as we will show, we can also benefit from this situation by exploiting collaborative effects and profit from changes that have occurred at other nodes in the system. Often one can benefit from having a community of users which allows for recommending relevant concepts according to similar interests. Of particular interest are therefore collaborative filtering systems which can produce personal recommendations by computing the similarity between own preferences and the ones of other people.

In our work we adapt a collaborative filtering recommender system to assist users in the management and evolution of their personal ontology by providing detailed suggestions of ontology changes. We illustrate the model for recommending ontology changes, the recommender method itself and its functionality in Section 9.1. The approach has been implemented as an extension of the Bibster application. We describe these extensions in Section 9.2. The evaluation of the approach was performed as an experiment within the Bibster community and shows very promising results, which we present in Section 9.3. In Section 9.4 we present related work in the areas of work in recommender systems, work in using taxonomies in recommender systems, and work in learning taxonomies and ontology evolution in general. Finally, we conclude and summarize in Section 9.5.

9.1 A Model for Recommending Ontology Changes

In this section we introduce our formal model for recommending ontology changes. This model directly builds on the definitions of the OWL ontology language presented in Definition 2 from Section 6.2 and the definitions of ontology change operations in Definition 14 from Section 8.2. Recall that we defined an ontology change operation $\text{oco} \in \text{OCO}$ as a function $\text{oco} : \mathcal{O} \rightarrow \mathcal{O}$, where OCO denotes the set of all change operations.

A recommender system for ontology changes tries to suggest ontology changes to the user based on some information about him and potential other users. Formally, an *ontology recommender* then is a function

$$\varrho : \mathcal{X} \rightarrow 2^{\text{OCO}} \tag{9.1}$$

where $\mathcal{X}$ contains suitable descriptions of the target ontology and user.

For example, let recommendations depend only on the actual state of a user's ontology, i.e., $\mathcal{X} = \mathcal{O}$, where $\mathcal{O}$ denotes the set of possible ontologies. A simple ontology evolution recommender can be built by just evaluating some heuristics on the actual state of the ontology, e.g., if the number of instances of a concept exceeds a given threshold, it recommends to add subconcepts to this concept. But without any additional information, this is hardly very useful, as we would not be able to give any semantics to these subconcepts: we could recommend to further subdivide the concept, but not how, i.e., neither be able to suggest a suitable label for these subconcepts nor assertions of instances to them. We will call such an approach *content-based* to distinguish it from more complex ones.

Recommendation quality eventually can be improved by taking into account other users' ontologies and thereby establishing some kind of collaborative ontology evolution scenario, where each user keeps his personal ontology but still profits from changes of other users. Collaborative filtering tries to make automatic predictions about the interests

of a user based interests of other users. The interest is measured in terms of items that are being rated. The basic assumption is that users that have rated items similarly in the past, will also agree on ratings in the future. Predictions are calculated based either on correlation measures between users or between the items that are being rated.

Applied to the task of recommending ontology changes, the ontology elements are the items that are being rated by the user. The ratings, i.e. user feedback about ontology elements, are captured by the contextual annotations as introduced in Definition 11. Generally, we can distinguish between *explicit* and *implicit* user feedback from usage information. We talk about explicit feedback if we allow that a user (i) can directly express how important a certain ontology element is for him, and that he (ii) can explicitly express negative ratings for ontology elements that he does not want to be part of his ontology.

We can obtain implicit feedback from log information that indirectly indicate the importance of ontology elements based on how they have been used. For example, we can use an implicit usage context called $r^u : N \rightarrow \mathbb{N}$, which indicates the relevance of the elements based on how they have been used, e.g. counts the number of queries issued by the user and instances in his knowledge base that reference a given symbol name. This information is available in a wide range of application scenarios. Of course, in specific scenarios further information may be available and thus additional contextual annotations can be defined.

In particular, we define the following two contextual annotations:

1. We use an explicit rating, called the membership-rating r^m with taboos, for which (i) all symbols and axioms actually part of the ontology have rating +1, (ii) all symbols and axioms not actually part of the ontology can be explicitly marked taboo by the user and then get a rating -1.

2. We use an implicit, usage-based rating called r^u, which indicates the relevance of the elements based on how it has been used. This relevance can be obtained from the percentage of queries issued by the user and instances in his knowledge base that reference a given symbol name.

As a correlation measure, we rely on the similarity between the users' ontologies. The basic idea is as follows: assume that for a target ontology we know similar ontologies called *neighbors* for short, then we would like to spot patterns in similar ontologies that are absent in our target ontology and recommend them to the target ontology. Another wording of the same idea is that we would like to extract ontology change operations that applied to the target ontology increases the similarity with its neighbors.

Let

$$\text{sim} : \mathcal{O} \times \mathcal{O} \rightarrow [0, 1] \tag{9.2}$$

be such a similarity measure – as introduced in the similarity framework in Chapter 7 – where $\mathrm{sim}(O, P)$ is large for similar ontologies O and P and small for dissimilar ontologies. Recall that ontologies in our scenario may have additional contextual annotations that are valuable information to consider in similarity measures suitable for recommendation tasks.

We can choose a simple correlation measure[1] (vector similarity) to compute similarities between ontologies of two users based on their ratings of the elements in the ontology:

$$\mathrm{sim}_r(O, P) := \frac{\sum_{s \in N} r_O(s)\, r_P(s)}{\sqrt{\sum_{s \in N} r_O(s)^2}\sqrt{\sum_{s \in N} r_P(s)^2}} \tag{9.3}$$

Similarities for the two different rating annotations r^m and r^u are computed separately and then linearly combined with equal weights:

$$\mathrm{sim}(O, P) := \frac{1}{2}\,\mathrm{sim}_{r^m}(O, P) + \frac{1}{2}\,\mathrm{sim}_{r^u}(O, P) \tag{9.4}$$

Finally, as in standard user-based collaborative filtering, ratings of all neighbors Ω are aggregated using the similarity-weighted sum of their membership ratings r^m for a given concept c, allowing for a personalized recommender function:

$$r_{\mathit{personalized}}(O, \Omega, c) := \frac{\sum_{P \in \Omega} \mathrm{sim}(O, P)\, r^m_P(c)}{\sum_{P \in \Omega} |\,\mathrm{sim}(O, P)|} \tag{9.5}$$

The recommendations are obtained directly from the rating: Elements with a positive rating are recommended to be added to the ontology, elements with a negative rating are recommended to be removed. Disregarding the similarity measure between the users' ontologies, we can build a naive recommender that does not provide personalized recommendations, but instead simply identifies "most popular" operations based on an unweighted average of the membership ratings:

$$r_{\mathit{baseline}}(O, \Omega, c) = \frac{\sum_{P \in \Omega} r^m_P(c)}{|\Omega|} \tag{9.6}$$

9.2 Recommending Ontology Changes in the Bibster System

In this section we will first describe the role of personalized ontologies in our bibliographic application scenario. We will then describe how the recommender functionality is applied in the system to support the users in evolving their personalized ontologies.

[1] Please note that depending on the range of the rating annotations in question, the correlation measure may need to be normalized to [0,1].

Extensions for Evolution and Recommendations. In Bibster we initially assumed both the application (SWRC) and domain (ACM) ontologies to be globally shared and static. This basically holds for the application ontology, but users want to adapt the domain ontology continuously to their needs. This is largely motivated by the sheer size of the ACM Topic Hierarchy which makes browsing, and therefore also querying and manual classification, difficult for users.

As part of this work we implemented extensions as described in the previous Section 9.1 to Bibster which support the evolution – i.e. the continuous adaptation – of the domain ontology by the users. A basic assumption here is that all users agree in general on the ACM Topic Hierarchy as domain ontology, but each user is only interested in seeing those parts of it which are relevant for him at a certain point of time.

In the application, we have separated the interaction with the ontology in two modes: a *usage mode* and an *evolution mode*. The usage mode is active for the management of the bibliographic metadata itself, i.e. creating and searching for the bibliographic metadata. This mode only shows the current view on the ontology consisting of the topics that the user has explicitly included in his ontology. The evolution mode allows for the adaptation of the ontology. In this mode also the possible extensions along with the corresponding recommendations are shown.

Ontology Change Operations To keep things simple and trying to separate effects from different sources as much as possible, we allow as change operations the addition and removal of topics from the personal ontology. More specifically, this addition/removal corresponds to the addition/removal of the individual assertion axiom (e.g. $Topic(Knowledge\_Representation\_Formalisms)$ and the role assertion axiom that fixes the position in the topic hierarchy (e.g. $SubTopic(Artificial\_Intelligence, Knowledge\_Representation\_Formalisms)$). The addition of topics is restricted to those topics that are predefined in the ACM Topic Hierarchy. Also, the position of the topics is fixed globally by the background ontology.

Ontology Ratings To elicit as much information as possible from users' work with the application, we gather various ontology rating annotations in the different modes.

We obtain the membership-rating r^m in the evolution mode from the explicit user actions (c.f. Figure 9.2): The user can either add a topic in the taxonomy, which will assigning a rating +1 for the topic, or he can exclude (taboo) the topic from the taxonomy, which will assign -1 for the explicitly taboo-ed topic.

We obtain the usage-based rating r^u in the usage mode by counting the percentage of queries issued by the user and instances in his knowledge base that reference a given topic. (For this, references to all topics are retained, especially also to topics not contained in the ontology of the user.)

The ontology ratings of the individual users are propagated together with peer profile descriptions as advertisements in the Peer-to-Peer network, such that every peers is informed about the usage of the ontology in the network. For the details of this process, we refer the reader to [HBE+04].

Recommending Ontology Changes For the recommendations of topics we rely on the rating function $r_{\text{personalized}}$ presented in the previous section. From the ratings of the topics, we can directly obtain the recommendations: Topics with a positive rating are recommended to be added to the ontology, topics with a negative rating are recommended to be removed. (Please note that adding a topic actually means adding the corresponding axioms, as described above.)

Topics in the topic hierarchy are visualized depending on the current rating r^m of the topic and on the recommendation for the topic using a the coding scheme shown in Figure 9.1. Intuitively, $+$ and $-$ denote recommendations to add and remove a topic,

	Recommendation		
Rating	Remove	Neutral	Add
Taboo-ed	X `topicname`	X `topicname`	**+** `topicname`
Unrated	**-** `topicname`	**?** `topicname`	**+** `topicname`
Accepted	**-** `topicname`	$\surd$ `topicname`	$\surd$ `topicname`

Figure 9.1: Visualization of Topics in Evolution Mode

respectively, $\surd$ and X denote already accepted or taboo-ed topics, where no opposite recommendation is present, and $?$ denotes an undetermined case. Figure 9.2 shows a screenshot of the ontology in the evolution mode.

9.3 Evaluation

For our evaluation, we wanted to study two questions: (i) do users accept recommendations for ontology changes at all? (ii) is a personalized recommender better suited for the task than a naive, non-personalized recommender?

To answer these questions, we have performed a user experiment in an in-situ setting using the Bibster system, in which we compared the baseline (non-personalized) and the personalized recommender, as defined in the previous section. In the following we will describe the setup of the experiment, evaluation measures, and the results.

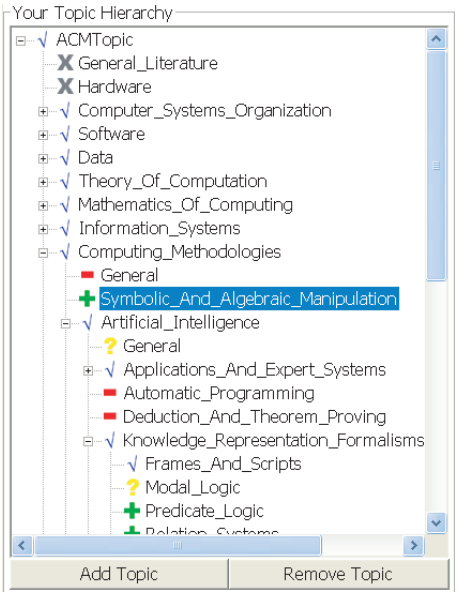


Figure 9.2: Screenshot of Recommended Topics

9.3.1 Design of the Experiment

The experiment was performed within three Computer Science departments at different locations. For a pre-arranged period of one hour, 23 users were actively using the system. The recommender strategy (baseline or personalized) was chosen randomly for each user at the first start of the Bibster application. The users were not aware of the existence of the different recommendation strategies.

During the experiment, the users performed the following activities (in no particular order), which are typical for the everyday use of the system:

- *Import data:* The users need to load their personal bibliography as initial dataset. This data should also reflect their research interest. As described before, the classification information of the bibliographic instances is part of the ontology rating and thus used to compute the similarity between the peers.

- *Perform queries:* The users were asked to search for bibliographic entries of their interest by performing queries in the Peer-to-Peer system. These queries might refer to specific topics in the ontology, and were thus again used as ontology ratings.

- *Adapt ontology:* Finally the users were asked to adapt their ontology to their personal needs and interests by adding or removing topics. This process was guided by the recommendations of the respective recommender function. The recommendations were updated (recalculated) after every ontology change operation.

The user actions were logged at every peer for later analysis. The logged information included: The type of the action (e.g. user query, ontology change operations), the provided recommendations, and a timestamp.

9.3.2 Evaluation Measures

We base our evaluation on the collected usage information in the form of events $\mathcal{E} \subseteq \text{OCO} \times 2^{\text{OCO}}$. An event is a tuple $(e, \hat{E})$ that consists of the actual user action $e \in \text{OCO}$, i.e., the specific ontology change operation performed, and the set $\hat{E} \subseteq \text{OCO}$ of recommendations at that point in time.

We observe a successful recommendation or a *hit*, when $e \in \hat{E}$, which means that the user performed an action that was recommended at that time. For non-hits, we distinguish two situations: (i) If the actual recommendation was exactly the opposite action, e.g., we recommended to add a topic but the user taboo-ed it, then we call this an *error*. (ii) If there was no recommendation for this action neither for its opposite, we call this *restraint*. Based on these counts, we can compute the following performance measures.

$$\textit{recall}(\mathcal{E}) := \frac{|\{(e,\hat{E}) \in \mathcal{E} \mid e \in \hat{E}\}|}{|\mathcal{E}|} \tag{9.7}$$

$$\textit{error}(\mathcal{E}) := \frac{|\{(e,\hat{E}) \in \mathcal{E} \mid \text{opp}(e) \in \hat{E}\}|}{|\mathcal{E}|} \tag{9.8}$$

$$\textit{restraint}(\mathcal{E}) := \frac{|\{(e,\hat{E}) \in \mathcal{E} \mid \text{opp}(e) \notin \hat{E} \wedge e \notin \hat{E}\}|}{|\mathcal{E}|} \tag{9.9}$$

where opp denotes the respective opposite operation, e.g., $\text{opp}(e^+) := e^-$ and $\text{opp}(e^-) := e^+$. Higher recall and lower error and restraint are better.

For a higher level of detail, we do so not only for all user actions, but also for some classes of user actions, such as all *add*- and all *remove/taboo*-operations.

As each of the measures alone can be optimized by a trivial strategy[2], we also computed the profit of the recommenders w.r.t. the profit matrix in Table 9.1:

$$\textit{profit}(\mathcal{E}) := \frac{\sum_{(e,\hat{E}) \in \mathcal{E}} \sum_{\hat{e} \in \hat{E}} \textit{profit}(e, \hat{e})}{|\mathcal{E}|} = \textit{recall}(\mathcal{E}) - \textit{error}(\mathcal{E}) \tag{9.10}$$

An intuitive reading of the profit is: The higher the profit, the better the performance of the recommender. In the best case ($profit = 1$), all user actions were correctly recommended by the system, in the worst case ($profit = -1$), all user actions were opposite of the recommendation.

$profit(\mathcal{E})$	Recommendation		
User Action	Remove	None	Add
Remove	1	0	-1
None	0	0	0
Add	-1	0	1

Table 9.1: Evaluation Profit Matrix

ACM Topic	# Add Actions
Information_Systems	23
Computing_Methodologies	15
Data	14
Computing_Methodologies/Artificial_Intelligence	12
Information_Systems/Database_Management	12
Software	11
Mathematics_Of_Computing	10
Computer_Systems_Organization	10
Computer_Systems_Organization/Computer_Communication_Networks	10
Computing_Methodologies/Artificial_Intelligence/ Knowledge_Representation_Formalisms_And_Methods	10

Table 9.2: Most Popular Topics

9.3.3 Evaluation Results

For the 23 participating users in the experiment, the baseline recommender was active for 10 users, the personalized recommender was active for the other 13 users. The participants performed a total of 669 user actions (452 add topic and 217 remove topic), 335 of these action were performed by users with the baseline strategy, 334 by users with the personalized recommender. Table 9.2 shows the number of add-topic-actions for the most popular topics. Figure 9.3 shows the cumulative results of the performance measures defined above for the baseline and the personalized recommender. The diagrams show the results for *Add* and *Remove* operations separately, as well as combined for all change operations.

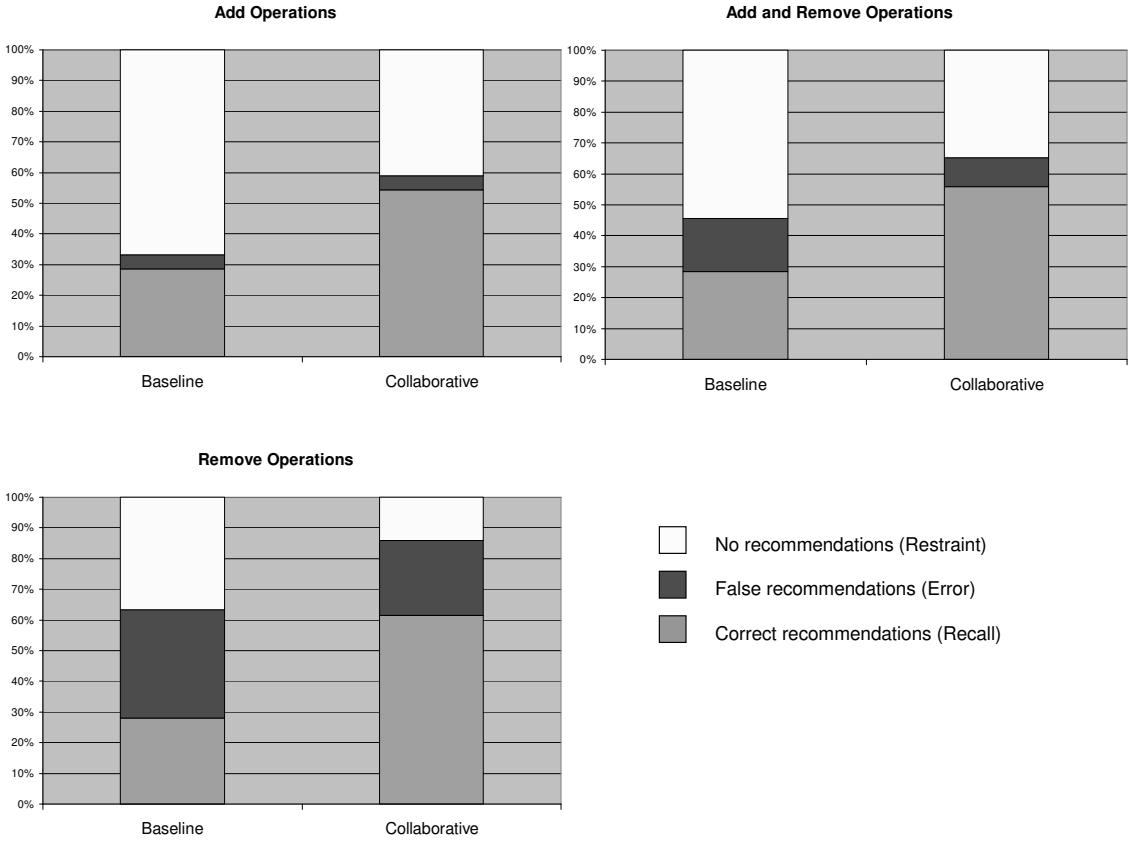


Figure 9.3: Performance Measures of the Recommender

As we can see in Figure 9.3 (upper right), overall the personalized recommender correctly recommended more than 55% of the user actions, while the baseline achieved less than 30%. The error rate of the baseline algorithm is considerably higher: We observed

[2]For example, to maximize the recall of add operations, one would trivially recommend to add all topics.

an *error* $= 17\%$ and 9% for the baseline and the personalized approach, respectively. Further we observed a very large amount of restraint operations with *restraint* $= 67\%$ for users with the baseline strategy. Probably this is the result of a large number of recommendations irrelevant to the user given by the system with the baseline strategy. In such a case the user would not like to follow the system and constructs the ontology mainly by themselves. Only from time to time he takes some of the recommendations into account.

By comparing add and remove operations we observe a higher amount of *error* recommendations for remove operations in comparison to the a really small amount of it for the add recommendations while the correct recommendations are comparable for both operations (cf. Figure 9.3, left side). We think that this observation is based on the fact that a user is more likely to follow an add operation without a "substantiated" reason or explanation than a remove operation. While adding something to his "collection" and following the idea of having more the remove operation forces the feeling of "loosing" something, so typically users are more reluctant to remove topics.

Calculating the overall profit of the two recommender functions, we obtain *profit*$(\mathcal{E}) = 0.11$ for the baseline recommender. For the collaborative recommender, we obtain a significantly better value of *profit*$(\mathcal{E}) = 0.47$. Concluding we can state that the personalized recommender function provides substantially more useful recommendations.

9.4 Related Work

Related work exists in three different aspects: work in recommender systems, especially collaborative filtering in general, work in using taxonomies in recommender systems, and work in learning taxonomies and ontology evolution in general.

Recommender systems have their roots in relevance feedback in information retrieval [Sal71], i.e., adding terms to (query expansion) or re-weighting terms of (term re-weighting) a query to a document repository based on terms in documents in the result set of the original query that have been marked relevant or non-relevant by the user, as well as adaptive hypertext and hypermedia [SF91], i.e., the automatic adaptation of the link structure of a document repository based on previous link usage by users.

Although most recommender systems research meanwhile focuses on more complex models treating the task as a learning or classification problem, collaborative filtering models still are under active investigation [HKTR04, DK04] due to their simplicity and comparable fair quality.

Taxonomies are used in recommender systems to improve recommendation quality for items, e.g., in [MSR04] and [ZSTL04], where ontological inference is shown to improve user profiling, external ontological knowledge is used to successfully bootstrap a recommender system and profile visualization is employed to improve profiling ac-

curacy. But to our knowledge there is no former approach for the inverse task, to use recommender systems for the personalization of the taxonomy or more generally of an ontology.

Change discovery as an important aspect of ontology evolution has been addressed for example in [SMMS02]. One approach for *usage-driven change discovery* in ontology management systems has been explored in [SS02], where the user's behavior during the knowledge providing and searching phase is analyzed. [SHG03] describes a tool for guiding ontology managers through the modification of an ontology based on the analysis of end-users' interactions with ontology-based applications, which are tracked in a usage-log. These approaches are based on the identification of problems or anomalies in an existing ontology. On the contrary, in our work we identify positive parts in ontologies, which might be useful to introduce in the ontologies of other users. Further, the prior existing work only addressed the evolution of a single ontology in a centralized scenario. In our work we are extending the idea of applying usage-information to a multi-ontology model by using collaborative filtering to recommend ontology changes based on the usage of the personal ontologies.

9.5 Conclusions

We have presented an approach to recommend ontology change operations to a personalized ontology based on the usage information of the individual ontologies in a user community. We have adapted a collaborative filtering algorithm to determine the relevance of ontology change operations based on the similarity of the users' ontologies. The virtue of this approach lies in the fact that the characteristics of changes to multiple, distributed ontologies are exploited, such that an individual user can profit from changes other users have made.

In our experimental evaluation with the Peer-to-Peer system Bibster we have seen that the users actually accept recommendations of the system for the evolution of their personal ontologies. The results further show the benefit of exploiting the similarity between the users' ontologies in a personalized recommender compared with a simple, non-personalized baseline recommender.

In our experiment we have made various simplifying assumptions. Their relaxation will open fruitful directions for future work: We assumed a fixed background ontology which limits the space of change operations. Relaxing this assumption will introduce challenges related to aligning heterogeneous ontologies. Further, the recommendation of adding or removing concepts in a given concept hierarchy can only be a first step. Next steps will therefore also include recommendations of richer change operations.

Part IV

Ontology-based Coordination

Chapter 10

A Metadata Ontology for Peer-to-Peer Coordination

Coordination is the fundamental problem in distributed information systems that arises from the autonomy of nodes. The coordination problem is that of managing the organization of the interaction between nodes. In the case of completely centralized architectures, one node has complete control over all other nodes, whereas in completely decentralized architectures there is no central control and all nodes act autonomously. As a result of this autonomy, new coordination models are required that scale with the size of the distributed information system in terms of the number of interacting nodes. These coordination models need to take into account the various aspects and dimensions of autonomy: *Design autonomy* requires to coordinate heterogeneous information models, *communication autonomy* involves resource discovery and selection as well as routing of requests, and finally *execution autonomy* calls for coordination models that allow the execution of local operations without external interference from other nodes.

One important concept to deal with the coordination of a distributed information system is the concept of metadata. Metadata is explicitly managed data about the system elements to support their interoperation and coordination. In the absence of centralized control it is critical that resources to be managed are self-descriptive such that their properties, capabilities, and requirements can be interpreted consistently. In this chapter we present an ontology for the representation of such metadata. We also show how the various aspects of coordination can be addressed using this metadata ontology. In this approach, nodes – or peers – advertise descriptions of their resources and can thus establish acquaintances with other nodes. Acquainted peers can then share data and coordinate their interaction.

This chapter is organized as follows: In Section 10.1 we analyze the dimensions of autonomy and their consequences for coordination. In the subsequent sections we introduce the individual modules of the metadata ontology for Peer-to-Peer coordination: We discuss ontology metadata in Section 10.2 and peer metadata in Section 10.3. We then

present applications for the decentralized management of ontology metadata in Section 10.4. Finally, we discuss related work in Section 10.5 and conclude in Section 10.6.

10.1 Peer-to-Peer Coordination

In this section, we analyze the different types of autonomy in distributed information systems and resulting requirements for a metadata ontology for Peer-to-Peer coordination.

Design autonomy Existing approaches of ontology-based information access almost always assume a setting where information providers share an ontology that is used to access the information. In a Peer-to-Peer setting, this assumption does no longer hold. We rather face the situation, where individual peers maintain their own view of the domain in terms of the organization of the local file system and other information sources. Enforcing the use of a global ontology in such a setting would mean to give up the benefits of the Peer-to-Peer approach. Therefore, one has to find a way to deal with the existence of multiple, distributed and frequently changing views on the domain. Consequently, peers need to be able to describe the data they provide as well as the ontologies used to represent that data. Additionally, the semantic relationships and inter-dependencies between different ontologies need to be represented. Typically such coordination formulas rely on some mapping language, such as the one already presented in Chapter 6. Data coordination then involves the reconciliation and integration of data at query time or the maintenance of consistency in data contained within different peers.

Communication autonomy Communication autonomy means that peers are fully autonomous in choosing their acquaintances. Moreover, it is usually assumed that there is no global control in the form of a global registry to manage acquaintances. As a consequence, acquaintances need to be managed in a decentralized manner, i.e. by the individual peer. It is rather obvious that for a scalable coordination model it is critical how the acquaintances are established. While in an extreme case all peers are acquainted with either none or all other peers, an intelligent coordination model would establish acquaintances with "relevant" peers. An important question here is how to determine "relevance": A possible useful approach can base this definition on coverage of the shared data and ontologies as well as the expertise of peers. Communication autonomy also includes the freedom of peers to choose which information to provide to other peers: Some information may be of private nature and should not be visible to other peers.

Execution autonomy Execution autonomy refers to the ability of peers to perform local queries without the interference from other peers. An important assumption is that

a query is always defined with respect to the ontology of a single peer. Thus an individual peer needs only be aware, and knowledgeable, of the local ontology. In terms of execution, however, queries are classified into two categories. A local query is executed using only the data in the local peer, while a global query uses the Peer-to-Peer network to complement or reconcile locally retrieved data with data that resides in other peers. This requires the selection of relevant peer data sources as well as the establishment of the corresponding semantic relationships with the possibly heterogeneous data sources. Within Peer-to-Peer systems the availability of other peers is not always guaranteed. Moreover, some peers may have better connectivity, in terms of bandwidth, to the rest of the network than other peers. To improve network efficiency, data and ontologies may be duplicated on multiple local peers. The mechanisms for locating relevant replicas needs to be transparent to the user, but must be captured by the metadata model.

The metadata model we will introduce needs to reflect these requirements. The above analysis shows that the main entities of interest are the actors in the network, i.e. the peers, and the resources they provide, i.e. the ontologies. Correspondingly, we present in the following the two parts of the metadata model: ontology metadata and peer metadata.

10.2 Ontology Metadata

Ontology metadata is used to describe the primary informational resources being managed in a distributed information system. While it was already indicated in the prior section that we rely on metadata about ontologies, this choice for the granularity of the descriptions of the informational resources is not self-evident. For example, in the SWAP metadata model [EHvH+03] the granularity was chosen to be on the level of RDF statements. The major reason why that level of granularity was chosen was that the RDF ontology model does not provide any other model of granularity or modularization. It has turned out that this approach was too fine grained, the overhead of managing metadata on the level of single statements was too high, and as consequence it was rarely used. However, the OWL ontology model – which this work is based on – does provide a modularization model, in which ontologies are first class objects, which can be versioned, can import each other, etc. Consequently, we provide metadata on that level[1].

In order for metadata to be practically useful, two further aspects need to be considered.

1. The metadata needs to be agreed upon by the participating actors. It therefore is essential to agree on a *standard for the representation* of the metadata. Con-

[1]It should also be noted that this does still allow to provide metadata on the level of axioms if modules simply contain single axioms.

sequently we rely on OMV[2], the Ontology Metadata Vocabulary. OMV is the a proposed standard for ontology metadata that has been influenced by existing models such as the SWAP metamodel [EHvH+03] and OMO (Ontology Meta-Ontology) [MMS+03b]. It is currently being used in systems such as KAONp2p[3], Oyster [PH05] and Onthology [HPS+05] and is the agreed metadata standard in the Knowledge Web project[4].

2. While the ontology metadata vocabulary is meant to be applicable to a variety of applications, it needs to allow proprietary *extensions and refinements* for particular application scenarios. We therefore decided to design the OMV scheme modularly: OMV distinguishes between the OMV Core and various OMV Extensions. The former captures information which is relevant across application scenarios, whereas OMV extension modules are applicable to specific applications. One such extension is P-OMV for the description of peers providing the ontologies, which will be presented in the subsequent section.

In the following we provide an overview of the main properties of the OMV ontology with a focus on the properties relevant for the coordination problem in distributed information systems. The main classes and properties of the OMV ontology are illustrated in Figure 10.1. In the following we present a compact overview of the main properties of the class `Ontology`. For a complete reference and the complete ontology we refer the reader to `http://omv.ontoware.org/`.

As we have seen in the previous section, various types of metadata of ontologies need to be modeled in order to address the coordination problem, which we can classify as *Descriptive metadata*, *Provenance metadata*, *Dependency metadata*, and *Statistical metadata*.

Descriptive metadata Descriptive metadata relates to the domain modeled in the ontology in form of keywords, topic classifications, textual descriptions of the ontology contents etc. This type of metadata plays a crucial role in the discovery and selection of ontologies.

- `name`: The name by which the ontology is formally known. Typically the name is used for human interpretation.
- `language`: The language here refers to the natural language of the ontology, which is relevant for the applicability of the ontology as well as for the purpose of ontology mapping.

[2] `http://ontoware.org/projects/omv/`
[3] `http://ontoware.org/projects/kaonp2p/`
[4] `http://knowledgeweb.semanticweb.org/`

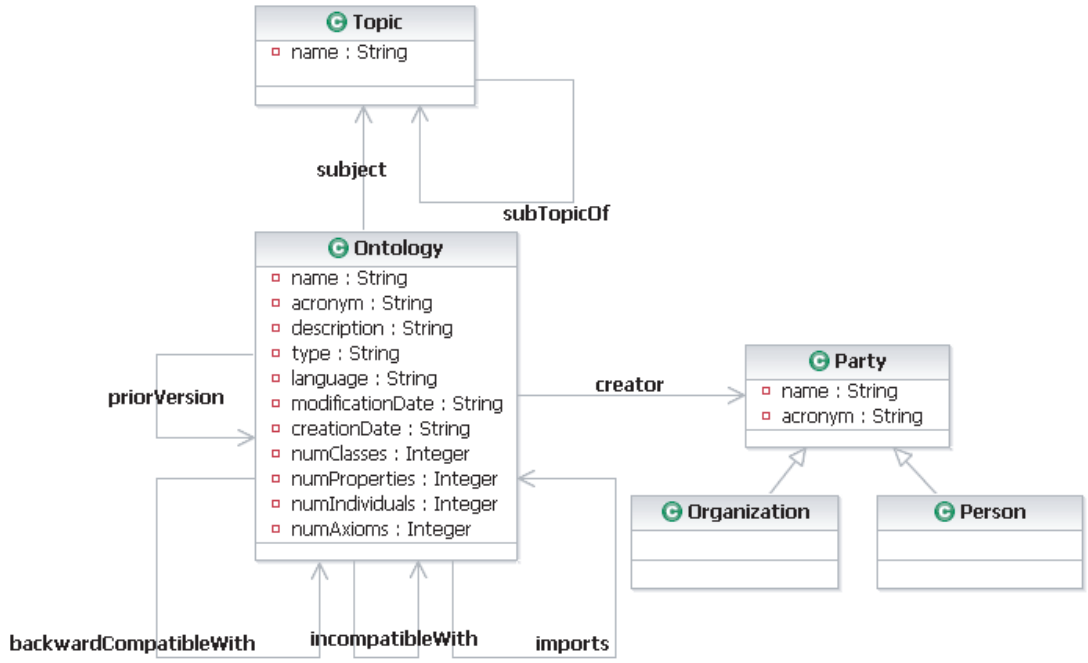


Figure 10.1: General OMV overview

- `type`: Ontologies may be categorized by different types of ontologies. This categorization can for example be based on the types identified in Section 3.1, i.e. *top-level*, *core*, *task*, *domain*, and *application ontology*.
- `subject`: The subject of an ontology provides a classification in terms of the domain. Typically, the subject is expressed as a classification against established topic hierarchies such as the general purpose topic hierarchy DMOZ[5] or the domain specific ACM topic hierarchy[6] for the computer science domain. The topics themselves may be organized in a topic ontology organized with relations such as `subTopicOf`.

Provenance metadata Provenance metadata provides information about the entities contributing to the creation of the ontology, as well as information about changes since its creation that are significant for its authenticity, integrity and interpretation.

- `creator`: The creator is the entity primarily responsible for producing the content of the ontology. The creator may be a party that is either a person or an organization.

[5] `http://dmoz.org/`
[6] `http://www.acm.org/class/`

- `creationDate` and `modificationDate` indicate when the ontology was first created and modified. The latter is especially important for the realization of caching and replication.

Dependency metadata Dependency metadata provides information to support managing relationships with other ontologies, such as dependencies on other ontologies, version compatibility, etc.

- `imports`: An `imports` statement references another ontology containing definitions, whose meaning is considered to be part of the meaning of the importing ontology. Importing another ontology brings the entire set of assertions provided by that ontology into the current ontology. Note that "imports" property is transitive; importing another ontology will also import all of the ontologies that the ontology imports.

- `backwardCompatibleWith`: A `backwardCompatibleWith` statement contains a reference to another ontology, which identifies the specified ontology as a prior version of a given ontology, and further indicates that it is backward compatible with it. In particular, this indicates that all identifiers from the previous version have the same intended interpretations in the new version.

- `incompatibleWith`: An `incompatibleWith` statement contains a reference to another ontology, which indicates that the given ontology is a later version of the referenced ontology, but is not backward compatible with it. Essentially, this means that the given ontology cannot be used in place of the referenced one without checking whether changes are required.

- `priorVersion`: A `priorVersion` statement contains a reference to another ontology, which identifies the specified ontology as a prior version of the containing ontology.

The reader may have noticed that the last four properties, which describe the dependency relationship with other ontologies, also exist as ontology properties in the OWL ontology model [BvHH+04]. These ontology properties constitute non-logical information, and certain constraints are put on their use that limit their applicability. In contrast, OMV metadata constitutes logical information. Furthermore, OMV also allows to describe ontologies in languages other than OWL. It is however desirable that the ontology metadata of OMV is extracted automatically from OWL annotation properties and ontology properties, if present.

Statistical metadata Statistical metadata relates to the size and type of the ontology. In particular we mention the number of specific ontological primitives (number

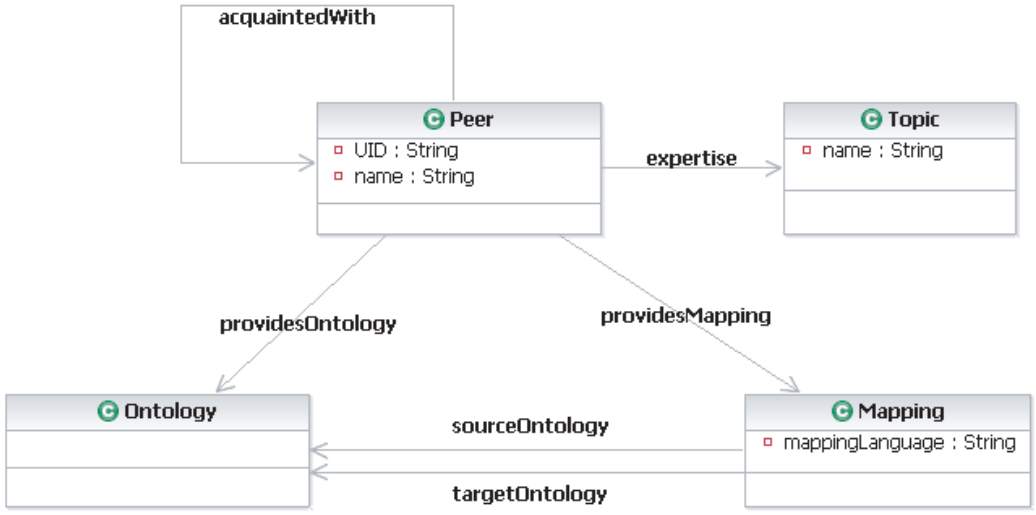


Figure 10.2: Overview of the P-OMV Ontology

of classes, properties, individuals, axioms). The availability of such metadata is important for assessing the quantity of information provided, as well as the type of information (e.g. large ABoxes vs. TBoxes), which can for example serve as rough indicators of the costs of processing the ontology.

10.3 Peer Metadata

Besides the ontology themselves, the second important resources to describe are the peers managing and providing these informational resources. The extensions required to model metadata of peers are realized as an extension to the OMV ontology, called P-OMV. Figure 10.2 shows an overview of the P-OMV ontology.

`Peer` The metadata required to describe peers include descriptive information about the peers themselves, their relationship with other peers, as well as information about the resources they provide:

- `UID`: Each peer has a unique ID to be identified. Depending on the underlying communication infrastructure, different addressing schemes may be applied. For example, in the Swapster architecure [EHvH+03], the JXTA UID is used to identify peers, as it relies on JXTA as the underlying communication infrastructure.

- `name`: In addition to the unique identifier, each peer carries a name for identification. In contrast to the UID, the name needs not to be unique in the network, as it is primarily used for human interpretation.

- `expertise`: The expertise is an abstract description of the peer in terms of some topic ontology. The expertise descriptions provide important information for the organization of the network and peer selection. Depending on the application scenario, the expertise of the peer can be the subjects of the ontologies that the peer provides, or a more generic description of expertise.

- `acquaintedWith`: This property describes the acquaintances of a peer with other peers. The Peer-to-Peer network then consists of local peers, each with a set of acquaintances, which define the Peer-to-Peer network topology. This topology may be dynamic, i.e. peers may be able to establish and modify acquaintances.

- `providesOntology`: This property describes the relationship between the peer and the ontologies provided by the peer. It is essential for locating relevant information resources in the network.

- `providesMapping`: This property is used to describe which mappings between ontologies a peer provides. The details of the Mapping class will be explained subsequently.

`Mapping` Mappings are used to describe the correspondences between different ontologies provided by the peers.

- `sourceOntology` and `targetOntology`: These properties specify the ontologies that are being mapped. In general, mappings need not be symmetric, a distinction between mapping source and target is therefore required.

- `mappingLanguage`: This property is used to indicate the language that is used to express the mapping. In Chapter 6 we have presented one particular formalism for ontology mappings, which can be expressed in SWRL. However, other languages may be used for the representation of ontology mappings.

Example 17 *Figure 10.3 shows a small example of the usage of P-OMV metadata. It depicts two acquainted peers that have an expertise in* Database Management *and* Information Systems. *They both provide an ontology about the bibliographic domain (SWRC and Proton*[7]*), which are related via a mapping provided by one of the peers.*

[7] `http://proton.semanticweb.org/`

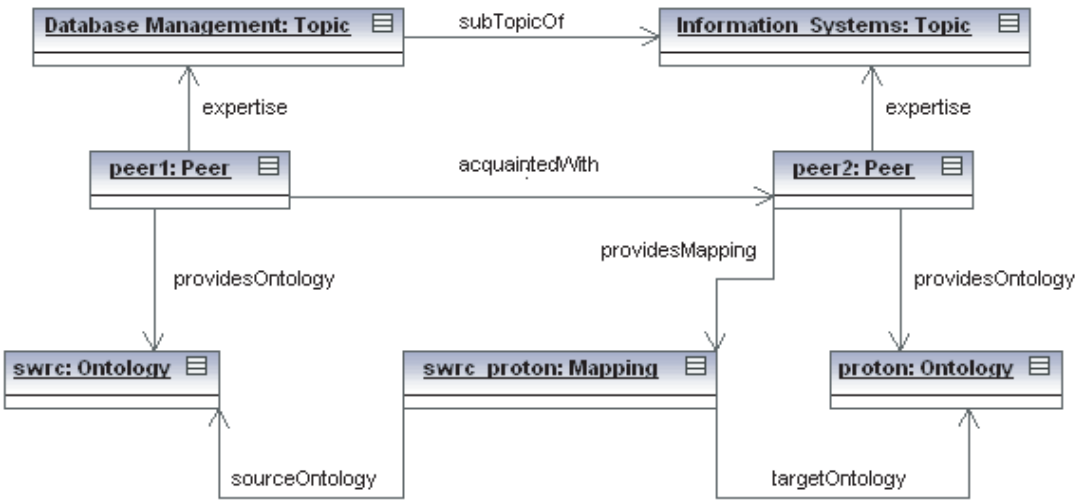


Figure 10.3: P-OMV Example Metadata

10.4 Managing Ontology Metadata in Distributed Information Systems

In the prior sections we have seen how resources in a distributed information system can be described with the OMV and P-OMV ontology. In order for the nodes to be able to act autonomously, it is essential for the nodes to be aware of the available resources in the system and to discover relevant resources as they become available. This in turn requires (1) registries for the management of the metadata descriptions and (2) a model for the propagation of metadata descriptions. Depending on the intended architecture of the distributed information system, there may be various solutions for a metadata registry with different degrees of decentralization. A completely centralized solution based on OMV has for example been developed in Onthology [HPS+05], where the entire metadata is managed on a single server. The model for propagation is simple, as all metadata descriptions are advertised to a single registry. Despite their simplicity, such completely centralized solutions are not applicable – or desirable – in all scenarios. In more decentralized environments, e.g. in true Peer-to-Peer applications, we require means to manage the acquaintances between peers without centralized control. We have developed such a completely decentralized solution with the Peer-to-Peer system Oyster, which we will explain in the following.

Oyster Oyster[8] [PH05] provides an infrastructure for storing, sharing and discovering ontologies in a decentralized manner. It can serve as a metadata registry that allows to access local metadata as well as metadata of remote peers in an integrated manner. Due to the lack of centralized control, there are only minimal administration efforts. The Oyster system has been implemented as an instance of the Swapster system architecture, which has already been introduced in Chapter 5. In addition to the infrastructure functionalities, Oyster provides a user interface, as shown in the screenshot in Figure 10.4. Oyster provides the relevant functionalities required for effective metadata management.

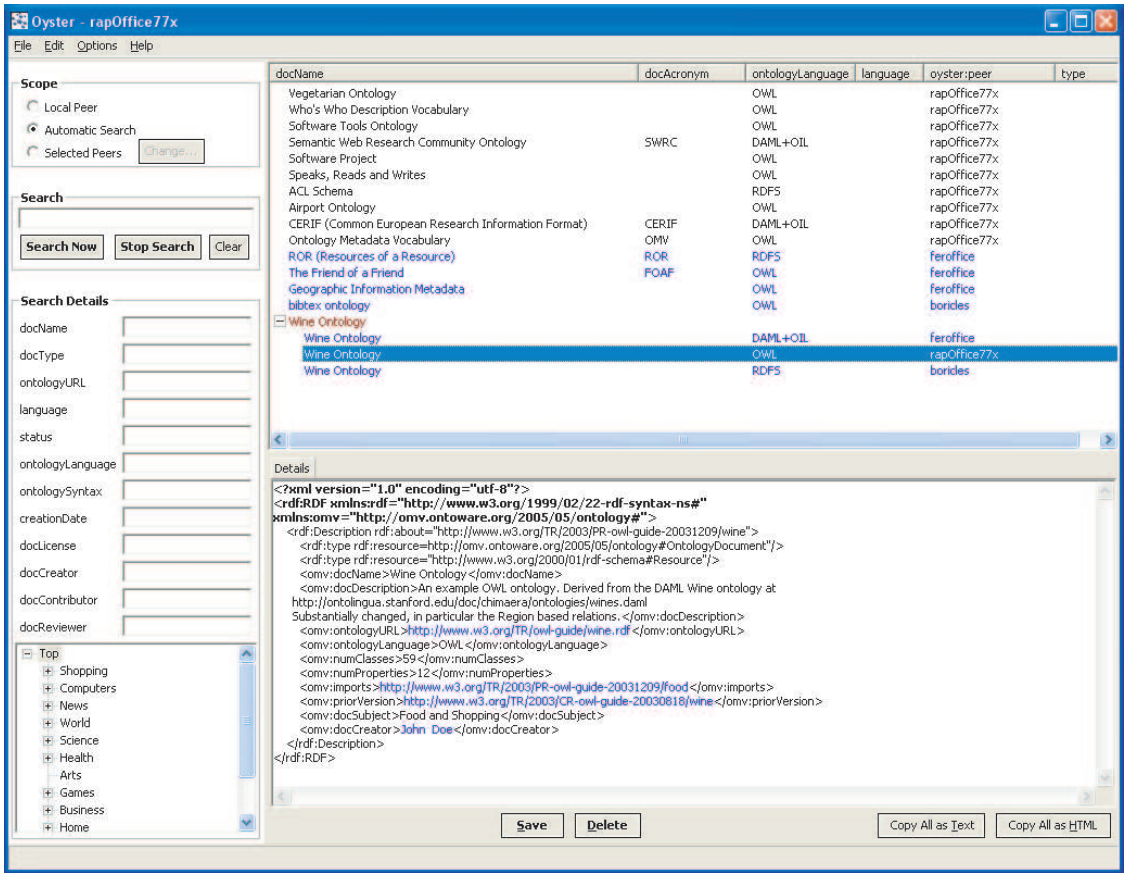


Figure 10.4: Oyster Screenshot

Creating and importing metadata: Oyster enables users to create metadata about ontologies manually, as well as to import ontology files and to automatically extract the ontology metadata available, letting the user fill in missing values. For the automatic extraction, Oyster supports the OWL, DAML+OIL and RDF-S ontology languages. The

[8] http://oyster.ontoware.org

ontology metadata entries are aligned and formally represented according to two ontologies: (1) the proposal for a metadata standard P-OMV which describes the properties of the ontology, (2) a topic ontology (e.g. the DMOZ topic hierarchy), which describes specific categories of subjects to define the domain of the ontology.

Searching for Ontologies: As shown in the left pane of the screenshot, the user can search for ontologies using simple keyword searches, or using more advanced, semantic searches. Here, queries are formulated in terms of the two ontologies (P-OMV and DMOZ). This means queries can refer to fields like name, acronym, ontology language, etc. (using the ontology document metadata ontology) or queries may refer to specific topic terms (using the topic hierarchy i.e. DMOZ).

Processing results: Finally, the results matching the query are presented in a result list (c.f. upper right pane in the screenshot). The details of particular results are shown in the lower right of the screenshot. The user can integrate results of a query into his local repository for future use. This information may in turn be used later to answer queries by other peers.

As we have mentioned above, the registry functionality for the management of metadata descriptions provided by Oyster is just one aspect relevant for the coordination of nodes in a distributed information system. The other important aspect is the model of how to propagate metadata descriptions and to automatically establish acquaintances between peers for an efficient organization of the network. These aspects of network organization will be discussed in detail in Chapter 11.

10.5 Related Work

The use of metadata for addressing coordination problems has a history in different communities. We discuss related work along the following categories: Metadata for ontologies, database coordination, and multi-agent coordination.

Metadata for ontologies In the past, there have been various proposals for modeling metadata of ontologies. Unfortunately, in the past neither has been accepted as a standard, some proposals, such as Dublin Core, were too general, others were limited in applicability. The Dublin Core (DC) metadata standard[9] is a simple yet effective element set for describing a wide range of networked resources. It includes two levels: Simple and Qualified. Simple DC comprises fifteen elements; Qualified DC includes an additional element as well as a group of element refinements (or qualifiers) that refine the semantics of the elements in ways that may be useful in resource discovery. [MMS+03b]

[9] http://dublincore.org/

has proposed an ontology meta-ontology (OMO) for a distributed ontology registry. Unfortunately, its use was limited to the KAON ontology model. The Semantic Web search engine SWOOGLE [DFJ+04] uses in particular the metadata that can be extracted automatically. Our approach includes and extends this metadata vocabulary. Ideally, future versions of SWOOGLE would also take into account the additional vocabulary defined in OMV. There exist some similar approaches to our proposed solution to share ontologies, but in general they are limited in scope. E.g. the DAML ontology library[10] provides a catalog of DAML ontologies that can be browsed by different properties.

Database Coordination The problem of database coordination has initially been studied in the context of database federations and more recently in the context of peer database management systems (PDBMS).

[SGMB03] introduces the Local Relation Model (LRM) as a data model specifically designed for Peer-to-Peer data management. LRM assumes that the set of all data in a Peer-to-Peer network consists of local (relational) databases, each with a set of acquaintances, which defines the Peer-to-Peer network topology. For each acquaintance link, domain relations define translation rules between data items, and coordination formulas defining semantic dependencies between the two databases. As such, LRM especially addresses the design autonomy, allowing for inconsistent databases and supporting semantic interoperability in the absence of a global schema.

In the Hyperion project [AKK+03], a PDBMS is envisioned as conventional DBMS augmented with a Peer-to-Peer interoperability layer, where this layer implements the functionality required for peers to share and coordinate data without compromising their own autonomy. The authors make use of different coordination rules for database coordination addressing different aspects of autonomy. The first set of coordination rules includes ones that manage consistency between the data of two peers by establishing semantic relationships between the schemas of peers, addressing the design autonomy. The second set of rules is created at query time, after an acquaintance has been established. They define the distributed execution model of the queries, respecting the execution autonomy of peers.

While in a sense these coordination rules specify the logical metadata that enables data sharing and coordination between independent, autonomous peers, there currently exist no proposals for a standard of such metadata that would allow consistent interpretation across applications.

Multi-Agent Coordination The coordination problem is one of the major problems studied in multi-agent systems. In contrast to distributed information systems, where the central concept is that of information, in agent systems the focus clearly is on coordinating *activities*. Nevertheless, the need for ontological approaches to describe agents,

[10] http://www.daml.org/ontologies/

their activities and resources has been recognized. A recent proposal of an ontology for dynamic coordination in multi-agent systems has been presented in [TAM+05]. This ontology focuses on agents, resources and activities as the main entities and the relationships and interdependencies between them. On the other hand, also in distributed information systems we see a gradual shift from concentrating on information-oriented approaches to service-oriented approaches. While these service-oriented aspects are not the focus of this work, first initial steps to modeling peers along with the services they provide have been presented in [HAS04].

10.6 Conclusions

The completely distributed nature and the high degree of autonomy of individual nodes comes with new challenges for the coordination of distributed information systems. Coordination models need to take into account the design-, communication-, and execution autonomy of nodes. In this chapter, we have presented a metadata ontology for semantic descriptions of resources in distributed information systems to support their interoperation and coordination. The metadata model we have described combines features of ontologies with rich metadata about the origin of the information. The metadata ontology builds on the core of the Ontology Metadata Vocabulary OMV, and extends it with the module P-OMV to describe peers as the providers of information.

A fundamental problem for the successful application of metadata is the question of how the metadata is generated. While certain metadata can be extracted from information sources automatically, some information may also have to be provided manually. It therefore is critical to support the creation, management and use of ontology metadata throughout the ontology lifecycle with the proper tools.

We have further presented Oyster, an application for the decentralized creation and management of ontology metadata. Oyster provides a registry for metadata descriptions based on OMV that allows to store, share and discover ontologies without any centralized control.

A key issue for the acceptance of using ontology metadata standards of course is a careful evaluation of the benefits of its use in general and in concrete applications. So far, OMV has proven to be useful in a variety of applications, including Oyster, Onthology and KAONp2p. In developing OMV we further relied on successful experiences with its predecessors, such as the SWAP metadata model, which has been applied in various applications of the SWAP system, including Bibster and Xarop. In the following chapter, we show the benefit of using the presented metadata model for a particular coordination problem: the problem of peer selection and query routing.

Chapter 11

Semantic Overlay Networks for Peer Selection

In the previous chapter we have identified different aspects of coordination problems in distributed information systems. In this chapter, we address one particular coordination problem, the one of resource discovery and selection. Resource discovery and selection is a well-known problem also in other forms of distributed systems and architectures, such as multi-agent systems, web services, etc. However, in this chapter we focus on Peer-to-Peer information sharing systems. In these systems, the problem amounts to that of peer discovery and peer selection: How do you find the right peers that are able to respond to your request in a large Peer-to-Peer system in a scalable manner without any centralized servers or hierarchy? This problem is at the heart of any Peer-to-Peer system. Peer-to-Peer networks that broadcast all queries to all peers do not scale – intelligent query routing and network topologies are required to be able to route queries to a relevant subset of peers. In particular, our approach of expertise based peer selection, relies on uses semantic overlay networks for efficient network organization.

In Section 11.1 we provide an overview of the landscape of existing approaches to network organization in Peer-to-Peer systems. In Section 11.2 we present our model of expertise based peer selection. In this model, peers use metadata descriptions – as introduced in the previous chapter – to advertise their expertise in the Peer-to-Peer network. Peers can thus establish acquaintances. The knowledge about the expertise of other peers forms a semantic overlay network, independent of the underlying network topology. If a peer receives a query, it can decide to forward it to acquainted peers whose expertise is *similar* to the subject of the query. The advantage of this approach is that queries will not be forwarded to all or a random set of known peers, but only to potentially relevant ones. We instantiate this model for the bibliographic scenario of the Bibster system in Section 11.3. In Section 11.4 we define evaluation criteria for our simulation experiments and a real-world field experiment. The results of the simulation experiments presented in Section 11.5 show how (1) the proposed model of expertise based peer selection consid-

erably improves the performance of the Peer-to-Peer system, (2) ontology-based matching with a similarity measure will improve the system compared with an approach that relies on exact matches, such as a simple keyword based approach, (3) the performance of the system can be improved further, if the semantic overlay network is built according to the semantic similarity of the expertise of the peers, (4) a "perfect" semantic overlay network imposed on the network using global knowledge yields ideal results. In Section 11.6 we present results from the field experiment with the Bibster system that validate the applicability and performance of the model for real-world systems. We discuss related work in Section 11.7 and conclude with some directions for future work in Section 11.8.

11.1 Approaches to Network Organization in Peer-to-Peer Systems

Peer-to-Peer systems are typically characterized by the absence of a single central instance of control. This has consequences for the network organization and the coordination to route requests to the peers able to respond to the request.

There exists a variety of approaches to network organization and peer coordination that stem from different communities, use different terminologies, but share similar concepts. This situation makes a crisp classification difficult. While we discuss specific approaches as part of the related work in Section 11.7, we here provide a general overview of Peer-to-Peer coordination that allows a placement of our approach. Figure 11.1 shows a tree-based view of the classification. It combines ideas of previous attempts to classifications of Peer-to-Peer systems [SAB+05, AH02].

Generally speaking, Peer-to-Peer networks are organized using so-called *overlay networks* (c.f. Figure 11.2) that serve as an abstraction layer over the underlying physical network, which is typically the Internet, i.e. TCP/IP based. In this overlay network peers connect with other peers according to some criteria. Here the most general distinction can be made according to the structure of the network into structured and unstructured networks. Structured networks are also referred to as Distributed Hash Tables (DHT) [SAB+05]. If the links in the network are created according to semantic relationships between the nodes, we speak of a semantic overlay network.

The approaches to the organization of the network and their indexing structure have a direct implication on the types of data structures and types of queries they support. For example, DHT-based approaches are limited to simple key-value lookups and do not support any more complex queries. On the other hand, completely unstructured approaches based on flooding can trivially broadcast arbitrary queries.

In the following we will discuss the classes of structured and unstructured networks as well as *semantic overlay networks* in more detail.

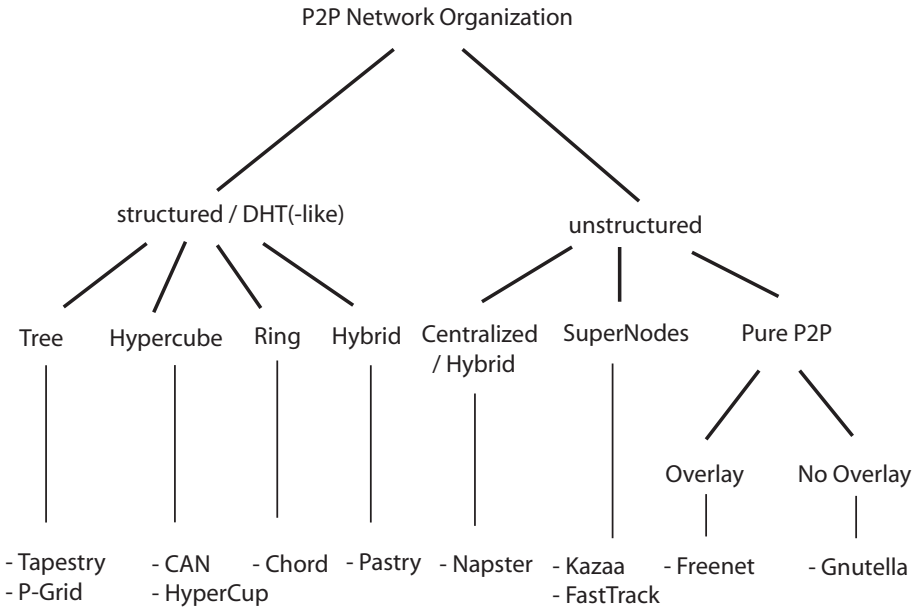


Figure 11.1: A classification of approaches to Peer-to-Peer network organization

11.1.1 Structured Networks

Distributed Hash Tables address a problem that is at the heart of many Peer-to-Peer systems – the lookup problem: How do you find any given data item in a large Peer-to-Peer system in a scalable manner, without any centralized servers or hierarchy? DHTs partition the ownership of a set of keys among participating nodes, and can efficiently route messages to the unique owner of any given key. Each node is analogous to a bucket in a hash table. DHTs are typically designed to scale to large numbers of nodes and to handle continual node arrivals and failures. In DHTs the nodes collectively form the system without any central coordination.

DHTs consist of two components: the *keyspace partitioning scheme* splits ownership of the keys among the nodes, and the *overlay network* connects the nodes and allows them to find the owner of a given key.

For the keyspace partitioning, most DHTs use some variant of consistent hashing to map keys to nodes. This technique relies on a function $d(k_1, k_2)$ which defines the distance from key k_1 to key k_2. Each node is assigned a single key called its identifier (ID). A node with ID i owns – that means it is responsible for – all the keys for which i is the closest ID, measured according to the distance function. Each node maintains a set of links to other nodes (its neighbors or routing table). Together these links form the overlay network, called the network topology. For any key k, a node either owns k or has a link to a node that is closer to k in terms of the keyspace distance defined above.

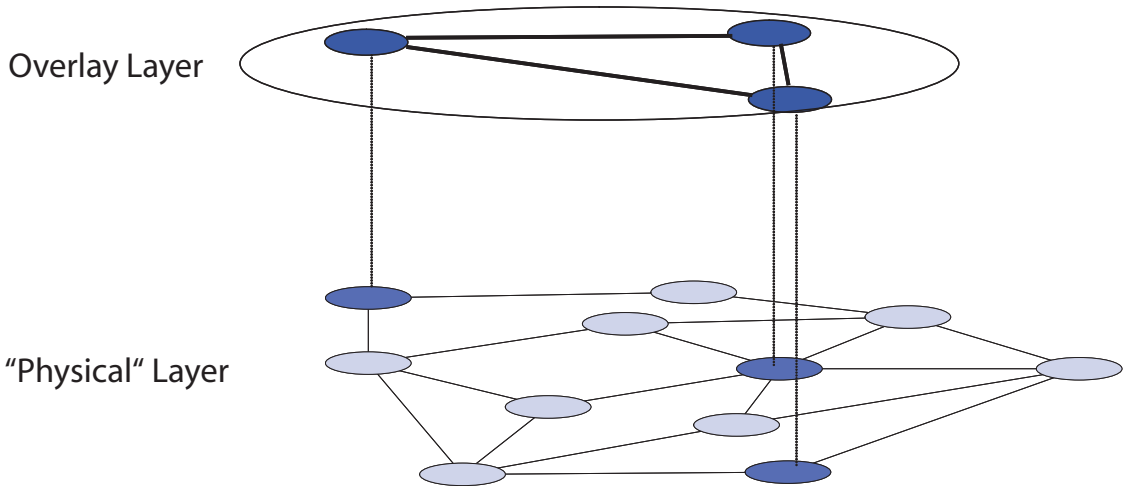


Figure 11.2: Overlay Networks

It is then easy to route a message to the owner of any key k using the following greedy algorithm: At each step, forward the message to the neighbor whose ID is closest to k. When there is no such neighbor, we have arrived at the closest node, which must be the owner of k as defined above. This style of routing is sometimes called key based routing.

Depending on the type of topology, we can further distinguish the different DHT approaches: In the tree-approach, the leaf nodes of the tree correspond to the node identifiers that store the keys to be searched. A popular DHT implementation that falls into this category is Tapestry [ZKJ02]. Chord [SMK+01] maintains a ring topology that resembles a skiplist. Hybrid topologies combine multiple of the above mentioned approaches, for example Pastry [RD01] uses a combination of a tree- and ring geometry. CAN [RFH+01] and HyperCup [SSDN02] support lookups in multiple dimensions: They use a d-dimensional Cartesian coordinate space to implement the DHT abstraction. The coordinate space is partitioned into hypercubes, called zones. Each node in the system is responsible for a zone, and a node is identified by the boundaries of its zone.

Typical DHTs allow to route content and queries in O(log(n)) steps to the right peers, where n is the number of peers in the network. Due to their deterministic structure and simple key-values mappings, they allow for perfect precision and recall. As DHTs essentially only provide a lookup operation, an inherent drawback is that they only directly support exact-match search: Objects that are not hashed cannot be found. However, additional functionality can be layered on top of a DHT. It is also important to note that the assigned responsibility of keys limits the autonomy of nodes in the sense that the assignment is beyond their decision and control.

Recently there have been proposals for the combination of the advantages of DHT-based approaches with those of unstructured networks, which will be explained subse-

quently. P-Grid [ACMD+03] is a Peer-to-Peer search system based on a virtual distributed search tree, similarly structured as standard Distributed Hash Tables. However, the structured overlay is constructed in a completely decentralized manner, just as in unstructured Peer-to-Peer networks. P-Grid is based on randomized algorithms for constructing the access structure, updating the data and performing search. At the same time, probabilistic estimates can be given for the success of search requests, and search is more robust than the previously described DHT approaches against failures of nodes. [ADHS05] presents recent results on a parallel and decentralized algorithm to construct structured overlays completely from scratch. This algorithm has been evaluated based on P-Grid, but is claimed to be applicable to any DHT approach.

11.1.2 Unstructured Networks

In our classification, we call unstructured networks those networks, that do not pre-impose a deterministic structure as in the case of DHTs. While this characterization is generally accepted [SAB+05], there also exist other characterizations, especially since many non-DHT based approaches also exhibit some kind of structure. In reality, there are varying degrees of structure, and it is a matter of perspective whether to call a system structured or not. For example, Aberer states in [ACMD+03]: "In unstructured P2P systems in principle peers are unaware of the resources that neighboring peers in the overlay networks maintain. Typically they resolve search requests by flooding techniques. Gnutella is the most prominent example of this class. In contrast, in structured P2P systems peers maintain information about what resources neighboring peers offer." According to this perspective, only the right-most path of the classification in Figure 11.1 would be considered unstructured.

However, for consistency we refer to DHT-based systems as structured networks, that pre-impose a deterministic structure, while in unstructured networks a possible structure is an emergent property.

In unstructured networks, we can also further classify along their topology, i.e. along the degree of central coordination or global knowledge. In the one extreme case, the coordination is managed with a centralized node, as for example in the case of Napster. In a hybrid case, there may be multiple of these centralized nodes. In a network with super nodes, the organization of the network is decentralized, but some nodes take over additional tasks. In the other extreme of a pure Peer-to-Peer system, no centralized or super nodes exist at all. Here we can further distinguish whether some sort of overlay network is built to route search requests, or whether simple flooding / broadcasting mechanisms are used.

Centralized / Hybrid Approaches An easy but not very robust approach is to have a centralized registry where peers can advertise their expertise descriptions or to have

the registry itself search the network for expertise descriptions. The most prominent example is Napster, a file-sharing application that relies on a centralized server with a directory of shared files stored on the individual peers. Such a repository can be seen as yellow pages, where each member in the network can look up the entity that fulfills its needs. In small organizations, approaches based on a single centralized registry could work very well because the network is small and stable, so that the registry does not have to do much query processing and updates. Often such approaches are called hybrid, as the lookup is performed in a centralized manner, whereas the actual data exchange is performed from peer to peer.

Super Nodes The approach based on super nodes (sometimes also called super peers) makes use of the different capacities of the nodes in a Peer-to-Peer network: Peers that have more processing power, memory or network bandwidth than other peers are assigned additional tasks in the network. One example of this approach that crosses centralized and decentralized topologies is FastTrack [SAB+05]. Its technology uses two tiers of control in the network. The first tier is made up of clusters of ordinary nodes that log onto super nodes with high speed connection. The second tier consists of only super nodes connected to one another in a decentralized fashion. The super nodes maintain routing tables, i.e. distributed indexes, which are used for resource discovery: When a node from the first tier makes a query, it is first directed to its own super node, who will in turn broadcast the query to all super nodes it is connected to. This is done repeatedly until a maximum number of hops is reached. FastTrack powers several successful Peer-to-Peer search engines, among them KaZaa client [SAB+05], a system which lets peers voluntarily act as super nodes.

Pure Peer-to-Peer Pure unstructured Peer-to-Peer systems abandon any sort of centralized control. In the simplest case, peers connect randomly with other peers, i.e. there is no additional overlay network used, and messages (or queries) are routed via broadcasting. Although a very simple technique, broadcasting has already proven its usefulness in small networks and in larger Peer-to-Peer file-sharing systems such as Gnutella [Kan99]. The Gnutella network uses a flat network of peers called servants (as they act as servers and clients) to maintain the index directory of all the content in the system. Servants are connected to each other in a flat ad-hoc topology that is dynamically kept alive by pinging known neighbors to discover new servants. When querying, a node simply broadcasts the query to its neighbors that keep forwarding the query to their neighbors until a sufficient number of answers is found or until maximum number of forwards (hops) are reached. In general, broadcasting approaches are not very scalable, because a query can result in a large number of messages which consumes an unacceptable usage of network capacity. Also it is possible that even if the data is somewhere in the network it will not be found due to the maximum number of hops. The main advantage of broad-

casting approaches is the low maintenance cost and dependency, meaning that almost no messages are needed to keep the network alive and that the network is very robust to frequent peer drops and joins (network dynamics).

To improve the performance of query routing, more advanced pure Peer-to-Peer approaches rely on overlay networks that allow to forward queries to peers based on some notion of relevance. The overlay network resembles a distributed index that is managed in a completely decentralized manner. The most prominent approch in this category is Freenet [CMH+02]. Freenet uses a key based routing mechanism: Each node maintains a data store containing documents associated with keys, and a routing table associating nodes with records of their performance in retrieving different keys. Queries are then routed to the peer with a key that is most similar to the requested one. In this sense, it is similar to key based routing in DHTs, but functions completely decentralized and non-deterministically. It is important to note that the similarity function operates on the keys, which are hash values over the content of the data, and therefore carries no notion of semantic closeness.

11.1.3 Semantic Overlay Networks

In contrast to approaches to network organization presented so far, Semantic Overlay Networks (SONs) are based on the idea that links between peers should be created based on *semantic relationships*. For example, an overlay could be formed in which peers maintain pointers to other peers with similar content. As such, similar nodes are clustered together, therefore SONs are sometimes also called Semantic Overlay Clusters, e.g. in [NWS+03]. As the name suggest, Semantic Overlay Networks are created *on top* of an underlying network layer, where peers are either connected randomly in the case of unstructured networks or based on the respective network topology in the case of structured networks. In a sense, the Semantic Overlay Network provides an additional layer of abstraction for an improved coordination of the peer system: It provides a flexible network organization that may improve query performance, while maintaining a high node autonomy. To our knowledge, the term Semantic Overlay Network has first been coined in [CGM02], however, the underlying ideas can be found in a variety of approaches.

Also the approach that we present in this chapter in the following sections falls into the category of Semantic Overlay Networks. As it does not require any form of centralization (pure Peer-to-Peer), i.e. we do not make any assumptions with respect to the structure of the underlying topology, it can theoretically be combined with any of the above introduced network structures.

Semantic overlay networks can be built with two different purposes: In one case, they can be used to impose a structure onto an initially unstructured network in order to organize the network and for example improve the performance of query routing. In a second case, they can be used ontop of existing structured networks, which already provide an efficient organization, in order to provide additional functionality beyond that

provided by the underlying layer. For example, in the case of DHTs, the only functionality provided is that of key-value lookups, which are performed in a very efficient manner. Additional functionality to support more complex queries and to deal with heterogeneous nodes can be built ontop.

11.2 A Model for Expertise Based Peer Selection

In the model we propose, peers advertise their expertise in the network to form acquaintances. The peer selection is based on matching the subject of a query and the expertise according to their semantic similarity. Figure 11.3 below shows the idea of the model in one picture.

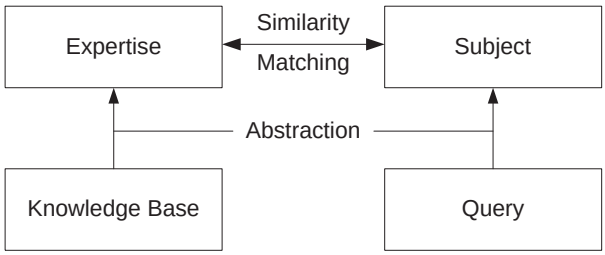


Figure 11.3: Expertise Based Matching for Peer Selection

In this section we first discuss how peers and their expertise are described and how peers promote their expertise as advertisement messages in the network. Second, we describe how the received advertisements allow a peer to select other peers for a given query based on a semantic matching of query subjects against expertise descriptions. The third part describes how a *semantic overlay network* can be formed by advertising expertise.

11.2.1 Semantic Description of Peers and their Expertise

For the semantic description of peers and their expertise we rely on the metadata ontology P-OMV presented in the previous chapter in Section 10.3. We here recapitulate the elements relevant for the process of peer selection.

Peers The Peer-to-Peer network consists of a set of peers P (captured by the P-OMV class `Peer`). Every peer $p \in P$ has a knowledge base described with an ontology that contains the knowledge that it wants to share (captured by the P-OMV property `providesOntology`).

Common Ontology The peers share a distinguished ontology O, which provides a common conceptualization of their domain by defining a set of concepts and the relations between them. The ontology is used for describing the expertise of peers and the subject of queries. In P-OMV, this common ontology amounts to the topic ontology, consisting of a set of topics T (corresponding to the P-OMV class `Topic`) and binary relations such as $subTopicOf \subseteq T \times T$.

Expertise An expertise description $e \in E$ is an abstract, semantic description of the knowledge base of a peer based on a set of terms of the common ontology O. Specifically, the expertise descriptions of P-OMV relate the peers with the terms of the topic ontology, which they are knowledgeable about via the property `expertise`. This expertise can either be extracted from the knowledge base automatically or specified in some other manner.

Advertisements Advertisements $A \subseteq P \times E$ are used to promote metadata descriptions of the peers and their expertise in the network. An advertisement $a \in A$ associates a peer p with an expertise description e. Peers decide autonomously, without central control, whom to promote advertisements to and which advertisements to accept. This decision can be based on the semantic similarity between expertise descriptions.

11.2.2 Matching and Peer Selection

We now turn to the discussion how peers are selected based on a given query using a similarity function to rank peers.

Queries Queries $q \in Q$ are posed by a user and are evaluated against the knowledge bases of the peers. First a peer evaluates the query against its local knowledge base and then decides which peers the query should be forwarded to. Query results are returned to the peer that originally initiated the query.

Subjects A subject $s \in S$ is an abstraction of a given query q expressed in a set of terms from the common ontology O. The subject can be seen as a complement to an expertise description, as it specifies the required expertise to answer the query.

Similarity Function We rely on the notion of a similarity function as defined in Definition 8 in the similarity framework for ontologies of Chapter 7: The similarity function $sim_S : S \times E \mapsto [0, 1]$ is a symmetric function that determines the semantic similarity between a subject $s \in S$ and an expertise description $e \in E$. As such, an increasing value indicates increasing similarity. If the value is 0, s and e are not similar at all; if the value is 1, they match exactly. sim_S is used for determining to which peers a query should be

forwarded. Analogously, the same kind of similarity function $sim_E : E \times E \mapsto [0,1]$ can be defined to determine the similarity between the expertise of two peers. The similarity function may be appropriately defined according to the semantics of the expertise and subject descriptions under consideration.

Peer Selection Algorithm The peer selection algorithm as shown in Algorithm 6 returns a ranked set of peers. The rank value is equal to the similarity value provided by the similarity function. From this set of ranked peers one can, for example, select the

Algorithm 6 Peer Selection

```
let A be the advertisements that are available on the peer
let γ be the minimal similarity between the expertise of a peer and the subject of the
query.
subject := ExtractSubject(query)
rankedPeers := ∅
for all ad ∈ A do
   peer := Peer(ad)
   rank := sim_S(Expertise(ad), subject)
   if rank > γ then
      rankedPeers := (peer, rank) ∪ rankedPeers
   end if
end for
return rankedPeers
```

best n peers, or all peers whose rank value is above a certain threshold.

11.2.3 Building a Semantic Overlay Network

The semantic overlay network is formed by establishing acquaintances between peers. These acquaintances essentially resemble the knowledge of the peers about the expertise of other peers in the network. It is important to state that this semantic overlay network is independent of the underlying network topology. At this point, we make no assumptions about the topology of the network.
The semantic overlay network can be described as a directed graph

$$(P, acquaintedWith)$$

where P is the set of nodes – or peers – and the edges are given by the following relation $acquaintedWith$:

$acquaintedWith \subseteq P \times P$, where $acquaintedWith(p_1, p_2)$ means that p_1 knows about the expertise of p_2.

The relation $acquaintedWith$ is established by the selection of which peers a peer sends its advertisements to and from which peers a peer accepts advertisements. The semantic overlay network in combination with the expertise based peer selection is the basis for intelligent query routing. The intuition of the overlay network is to establish acquaintances between peers with similar expertise in order to be able to route queries along a short path of increasing similarity between the subject of the query and the expertise of the peers. Different strategies for establishing such acquaintances will be presented and evaluated in the following sections.

11.2.4 Consequences of the Model

An important value of the model described above is that it dictates which design decisions must be made when introducing expertise based peer selection into a Peer-to-Peer network. The decisions are as follows:

- We must define the *common ontology* as a set of concepts and a set of relations between them.
- We must define *two abstraction functions*: one to abstract the contents of peers to expertise descriptions (sets of concepts from the ontology), and one to abstract queries to subjects (again sets of concepts from the ontology).
- We must define *two advertisement policies*: to which peers to send advertisements, and which advertisements to accept.
- We must define *two similarity functions*: one to compare subjects with expertise descriptions, and one to compare expertise descriptions with each other.
- We must define a *peer selection algorithm* to decide to which peers queries must be routed.

We believe this model to be of general value in understanding Peer-to-Peer models with semantic query routing. In the following section we instantiate this general model for our specific application scenario.

11.3 Peer Selection in the Bibster System

In this section we instantiate the general model for expertise based peer selection from the previous section for our bibliographic application scenario.

Peers A researcher is represented by a peer $p \in P$. Each peer provides a knowledge base, which consists of a set of bibliographic metadata items that are classified according to the ACM topic hierarchy[1].

Common Ontology The ontology O that is shared by all the peers is the ACM topic hierarchy. The topic hierarchy contains a set, T, of 1287 topics in the computer science domain and relations ($T \times T$) between them: *subTopic* and *seeAlso*.

Expertise The ACM topic hierarchy is the basis for our expertise model. Expertise E is defined as $E \subseteq 2^T$, where each $e \in E$ denotes a set of ACM topics, for which a peer provides classified instances.

Advertisements Advertisements associate peers with their expertise: $A \subseteq P \times E$. A single advertisement therefore consists of a set of ACM topics to which the peer is an expert.

Queries We use conjunctive queries as defined in Section 3.3 to express queries against knowledge bases of a peer. The following sample query asks for the titles of publications whose ACM topic is *Information Systems / Database Management*:

```
PREFIX swrc:  <http://swrc.ontoware.org/ontology#>
PREFIX acm:   <http://daml.umbc.edu/ontologies/topic-ont#>
PREFIX topic: <http://daml.umbc.edu/ontologies/classification#>

CONSTRUCT {?pub <swrc:title> ?title} FROM
{?pub rdf:type swrc:Publication;
  swrc:title ?title;
  acm:topic <topic:ACMTopic/Information_Systems/Database_Management>}
```

Subjects Analogously to the expertise, a subject $s \in S$ is an abstraction of a query q. In our scenario – where $S \subseteq 2^T$ – each s is a set of ACM topics, thus $s \subseteq T$. For example, the extracted subject of the query above would be $\{$ *Information Systems/Database Management*$\}$.

Similarity Function In this scenario, we use one similarity function sim ($sim = sim_E = sim_S$), which is based on the idea that topics which are close according to their positions in the topic hierarchy are more similar than topics that have a larger distance. For example, an expert on ACM topic *Information Systems/Information Storage*

[1] `http://daml.umbc.edu/ontologies/classification`

and Retrieval has a higher chance of giving a correct answer on a query about *Information Systems/Database Management* than an expert on a less similar topic like *Hardware/Memory Structures*.

To be able to define the similarity of a peer's expertise and a query subject, which are both represented as a set of topics, we first define the similarity for individual topics. [LBM03] have compared different similarity measures and have shown that for measuring the similarity between concepts in a hierarchically structured semantic network, like the ACM topic hierarchy, the taxonomic similarity (as introduced in Section 7) yields the best results:

$$S(t_1, t_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } t_1 \neq t_2, \\ 1 & \text{otherwise} \end{cases} \tag{11.1}$$

Here l is the length of the shortest path between topic t_1 and t_2 in the graph spanned by the *SubTopic* relation. h is the level in the tree of the lowest common subsumer from t_1 and t_2; $\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h, respectively. Based on benchmark data from [LBM03], the optimal values are: $\alpha = 0.2$, $\beta = 0.6$. Using the shortest path between two topics is a measure for similarity, as Rada and colleagues [RMBB89] have proven that the minimum number of edges separating topics t_1 and t_2 is a metric for measuring the conceptual distance of t_1 and t_2. The intuition behind using the depth of the direct common subsumer in the calculation is that topics at upper layers of hierarchical semantic nets are more general and are semantically less similar than topics at lower levels.

Now that we have a function for calculating the similarity between two individual topics, we define sim as:

$$sim(s, e) = \frac{1}{|s|} \sum_{t_i \in s} \max_{t_j \in e} S(t_i, t_j) \tag{11.2}$$

This function iterates over all topics t_i of the subject s and average their similarities with the most similar topic of the expertise e.

Peer Selection Algorithm The peer selection algorithm ranks the known peers according to the similarity function described above. Therefore, peers that have an expertise more similar to that of the subject of the query will have a higher rank. From the set of ranked peers, we now only consider a selection algorithm that selects the best n peers.

We have now made a decision on many of the points dictated by the general model from the previous section: a common ontology, expertise and query-subject descriptions,

advertisement-contents, and similarity functions. Still missing are the advertisement policy and the abstraction functions. These are experimental variables because we test different policies, and therefore will be discussed in Sections 11.5 and 11.6, where we describe the details of our experiments.

11.4 Evaluation Criteria

In this section we define a number of criteria for a Peer-to-Peer system, which will be the basis for the evaluation of our proposed model for peer selection. These criteria are mainly based on those described in [ESS+03]. We distinguish between input parameters that *affect* the performance of the system, and output parameters that *are affected* and serve as measures for the performance of the system.

11.4.1 Input parameters

The following input parameters are important criteria that affect the performance of a Peer-to-Peer system:

Number of Peers The size of the Peer-to-Peer network is represented by this number. Typically the scalability of the system is measured in terms of the number of peers.

Number of Documents The scalability of a Peer-to-Peer system can also be expressed in terms of the number of shared resource items, e.g. documents.

Document Distribution The document distribution in Peer-to-Peer networks is rarely completely random, but often has certain properties. With this input parameter we want to evaluate how the proposed model behaves with different document distributions.

Network Topology The performance of a Peer-to-Peer system is strongly influenced by the network topology and its characteristics. Possible topologies could for example be super-peer based, star or ring-shaped, or simply a random graph.

Advertisement Policy The advertisements are responsible for building the semantic overlay network. There are various variables involved, e.g. whom to send the advertisments to and which received advertisements to include based on the semantic similarity between the own expertise and that of the advertisement.

Peer Selection Algorithm The peer selection algorithm determines which peers a query should be forwarded to. This could be a naive algorithm, which simply broadcasts a query, or a more advanced one, as the proposed expertise based peer selection.

Maximum Number of Hops The maximum number of hops determines how many times a query is allowed to be forwarded. It determines how much the network will be flooded by a single query.

11.4.2 Output parameters

To evaluate a Peer-to-Peer system, we use precision and recall measures known from classical Information Retrieval. Here we distinguish measures on the document level (query answering) and the peer level (peer selection). These measures are defined as follows:

Document Level (Query Answering).

$Precision_{Doc} = \frac{|A \bigcap B|}{|B|}$
indicates how many of the returned documents are relevant, with A being the set of relevant documents in the network and B being the set of returned documents. In our model we work with exact queries, therefore only relevant documents are returned. The precision will therefore always be one:
$Precision_{Doc} = \frac{|B|}{|B|} = 1$.

$Recall_{Inf} = \frac{|A \bigcap B|}{|A|} = \frac{|B|}{|A|}$
The recall on the document level states how many of the relevant documents are returned.

Peer Level (Peer Selection).

$Precision_{Peer} = \frac{|A \bigcap B|}{|B|}$
For a given query, how many of the peers that were selected had relevant information. Here A is the set of peers that had relevant documents and B is the set of peers that were reached.

$Recall_{Peer} = \frac{|A \bigcap B|}{|A|}$
indicates for a given query, how many of the peers that had relevant information were reached.

Further Parameters. Another important output parameters is:

$Number_{Messages}$
: This output parameter indicates with how many messages the network is flooded by one query. The number of messages does not only affect the network traffic, but also CPU consumption, such as for the processing of the queries in the case of query messages.

There are many other output parameters that we could have used as additional evaluation criteria. Examples are the size of messages between peers, the response times on queries to the network, CPU load of individual peers etc. However, we do not report on these as they are not relevant to our evaluation hypotheses and therefore also not captured by our simulation software.

11.5 Simulation Experiments

In this section we describe the simulation of the bibliographic scenario presented in Section 11.3. The evaluations are based on the criteria defined in Section 11.4. With the experiments we validate the following hypotheses:

- **H1 - Expertise based selection:** The proposed approach of expertise based peer selection yields better results than a naive approach based on random selection. The higher precision of the expertise based selection results in a higher recall of peers and documents, while reducing the number of messages per query.
- **H2 - Ontology based matching:** Using a shared ontology with a metric for semantic similarity improves the recall rate of the system compared with an approach that relies on exact matches, such as a simple keyword based approach.
- **H3 - Semantic overlay network:** The performance of the system can be improved further, if the semantic overlay network is built according to the semantic similarity of the expertise of the peers. This can be realized, for example, by accepting advertisements that are semantically similar to the own expertise.
- **H4 - The "Perfect" overlay network:** Perfect results in terms of precision and recall can be achieved, if the semantic overlay network coincides with a distribution of the documents according to the expertise model.

11.5.1 Setup of the Simulation Experiments

In the following we describe the setup of the simulation experiments performed: the data sets used, the distribution of the data, the simulation environment, and the individual experimental settings.

Data Set To obtain a critical mass of bibliographic data, we used the DBLP data set, which consists of metadata for 380440 publications in the computer science domain (in the version from August 2003).

We have classified the publications of the DBLP data set according to the ACM topic hierarchy using a simple classification scheme based on lexical analysis: A publication is said to be about a topic, if the label of the topic occurs in the title of the publication. For example, a publication with the title "The Capabilities of Relational Database Management Systems." is classified into the topic *Database Management*.Topics with labels that are not unique (e.g. *General* is a subtopic of both *General Literature* and *Hardware*) have been excluded from the classification, because typically these labels are too general and would result in publications classified into multiple, distant topics in the hierarchy. Obviously, this method of classification is not as precise as a sophisticated or manual classification. However, a high precision of the classification is not required for the purpose of our simulations. As a result of the classification, about one third of the DBLP publications (126247 out of 380440) have been classified, against 553 out of the 1287 ACM topics. The classified DBLP subset has been used for our simulations.

Distribution of the Data We have simulated and evaluated the scenario with two different distributions, which we describe in the following. Note that for the simulation of the scenario we disregard the actual documents and only distribute the bibliographic metadata of the publications.

Topic Distribution: In the first distribution, the bibliographic metadata are distributed according to their topic classification. There is one dedicated peer for each of the 1287 ACM topics. The distribution is directly correlated with the expertise model, each peer is an expert on exactly one ACM topic and contains all the corresponding publications. This also implies that there are peers that do not contain publications, because not all topics have classified instances.

Proceedings Distribution: In the second distribution, the bibliographic metadata are distributed according to conference proceedings and journals in which the according publications were published. For each of the conference proceedings and journals covered in DBLP there is a dedicated peer that contains all the associated publication descriptions (in the case of the 328 journals) or inproceedings (in the case of the 2006 conference proceedings). Publications that are published neither in a journal nor in conference proceedings are contained by one separate peer. The total number of peers therefore is 2335 (=328+2006+1). With this distribution one peer can be an expert on multiple topics, as a journal or conference typically covers multiple ACM topics. Note that there is still a correlation between the distribution and the expertise, as a conference or journal typically covers a coherent set of topics.

Simulation Environment To simulate the scenario we have developed and used a controlled, configurable Peer-to-Peer simulation environment. A single simulation experiment consists of the following sequence of operations:

1. *Setup network topology:* In the first step we create the peers with their knowledge bases according to the document distribution and arrange them in a random network topology, where every peer knows 10 random peers. We have fixed this number in our simulations to keep the number of different variable tractable, and have chosen this value to simulate a realistic sparse topology. We do not make any further assumptions about the network topology.

2. *Advertising Knowledge:* In the second step, the semantic overlay network is created. Every peer sends an advertisement of its expertise to all other peers it knows based on the network topology. When a peer receives an advertisement, it may decide to store all or only selected advertisements, e.g. if the advertised expertise is semantically similar to its own expertise. After this step the semantic overlay network is static and will not change anymore.

3. *Query Processing:* The peers randomly initiate queries from a set of randomly created 12870 queries, 10 for each of the 1287 ACM topics. The peers first evaluate the queries against their local knowledge base and then propagate the query according to their peer selection algorithms described below.

Experimental Settings In our experiments we have systematically simulated various settings with different values of input variables. In the following we will describe an interesting selected subset of the settings to prove the validity of our hypotheses.

Setting 1 In the first setting we use a naive peer selection algorithm, which selects n *random* peers from the set of peers that are known from advertisements received, but disregarding the content of the advertisement. In the experiments, we have used n=2 in every setting, as a rather arbitrary choice.

Setting 2 In the second setting we apply the expertise based selection algorithm. The *best* n (n=2) peers are selected for query forwarding. Here the peer selection algorithm only considers *exact* matches of topics.

Setting 3 In the third setting we modify the peer selection algorithm to use the ontology based similarity measure, instead of only exact matches. The peer selection only selects peers whose expertise is equally or more similar to the subject of the query than the expertise of the forwarding peer.

Table 11.1: Experimental Settings for Simulation of Expertise Based Peer Selection

Setting #	Peer Selection	Advertisements	Topology
Setting 1	random	accept all	random
Setting 2	exact match	accept all	random
Setting 3	ontology based match	accept all	random
Setting 4	ontology based match	accept similar	random
Setting 5	ontology based match	accept similar	perfect

Setting 4 In the fourth setting we modify the peer to only accept advertisements that are semantically similar to its own expertise. The threshold for accepting advertisements was set to accept on average half of the incoming advertisements.

Setting 5 In this setting we assume global knowledge to impose a perfect overlay network on the peer network. In this perfect overlay network the acquaintances coincide with the ACM topic hierarchy: Every peer is acquainted with exactly those peers that are experts on the neighboring topics of its own expertise. This setting is only applicable for the distribution of the publications according to their topics, as it assumes exactly one expert per topic.

Table 11.1 summarizes the values of the input variables for the described settings.

11.5.2 Results

Figures 11.4 through 11.6 show the results for the different settings and distributions. The simulations have been run with a varying number of allowed hops. In the results we show the performance for a maximum of up to eight hops. Zero hops means that the query is processed locally and not forwarded. Please note that the diagrams for the recall and the number of messages per query (i.e. Figures 11.5, 11.6, 11.7) present cumulative values, i.e. they include the sum of the results for *up to* n hops. The diagram for the precision (Figure 11.4) of the peer selection displays the precision for a particular number of hops.

In the following, we interpret the results of the experiments for the various settings described above with respect to our hypotheses H1 through H4.

R1 - Expertise based selection The results of Figure 11.4, Setting 1, show that the naive approach of random peer selection gives a constant low precision of 0.03% for the topic distribution and 1.3% for the proceedings distribution. This results in a fairly

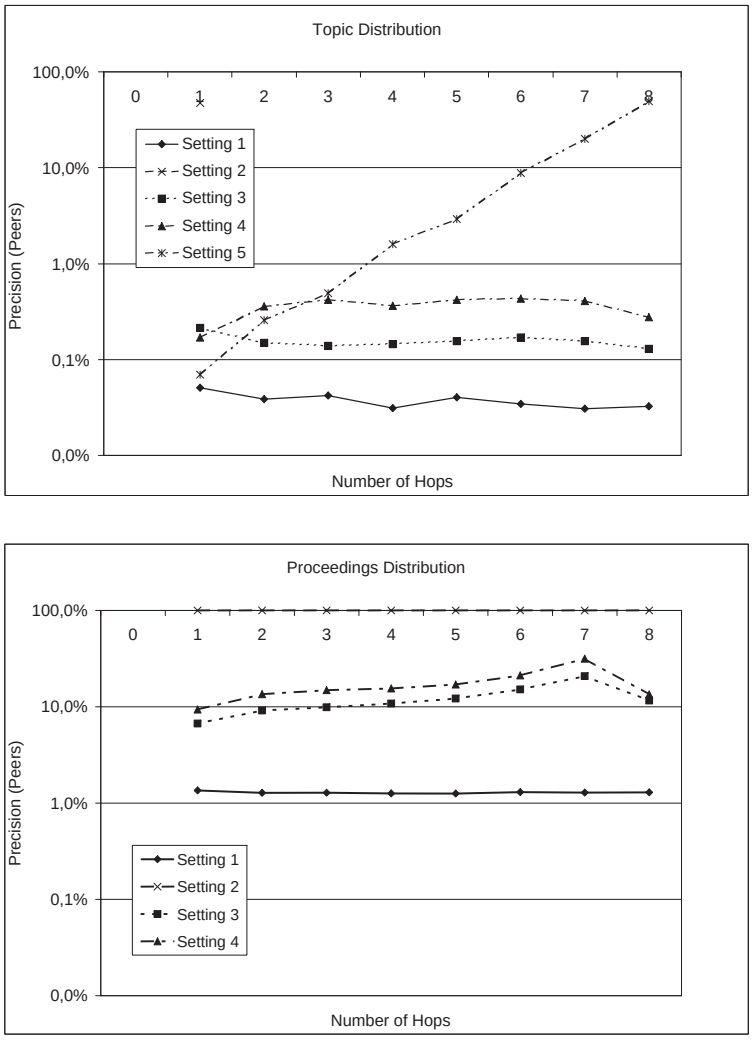


Figure 11.4: $Precision_{Peers}$ of Expertise Based Peer Selection

low recall of peers and documents despite a high number of messages, as shown in Figures 11.5, 11.6, 11.7, respectively. With the expertise based selection, either exact or similarity based matching, the precision can be improved considerably by about one order of magnitude. For example, with the expertise based selection in Setting 3, the precision of the peer selection (Figure 11.4) can be improved from 0.03% to 0.15% for the topic distribution and from 1.3% to 15% for the proceedings distribution. With the precision, also the recall of peers and documents rises (Figures 11.5, 11.6). At the same time, the number of messages per query can be reduced. The number of messages sent is influenced by two effects. The first effect is message redundancy: The more precise

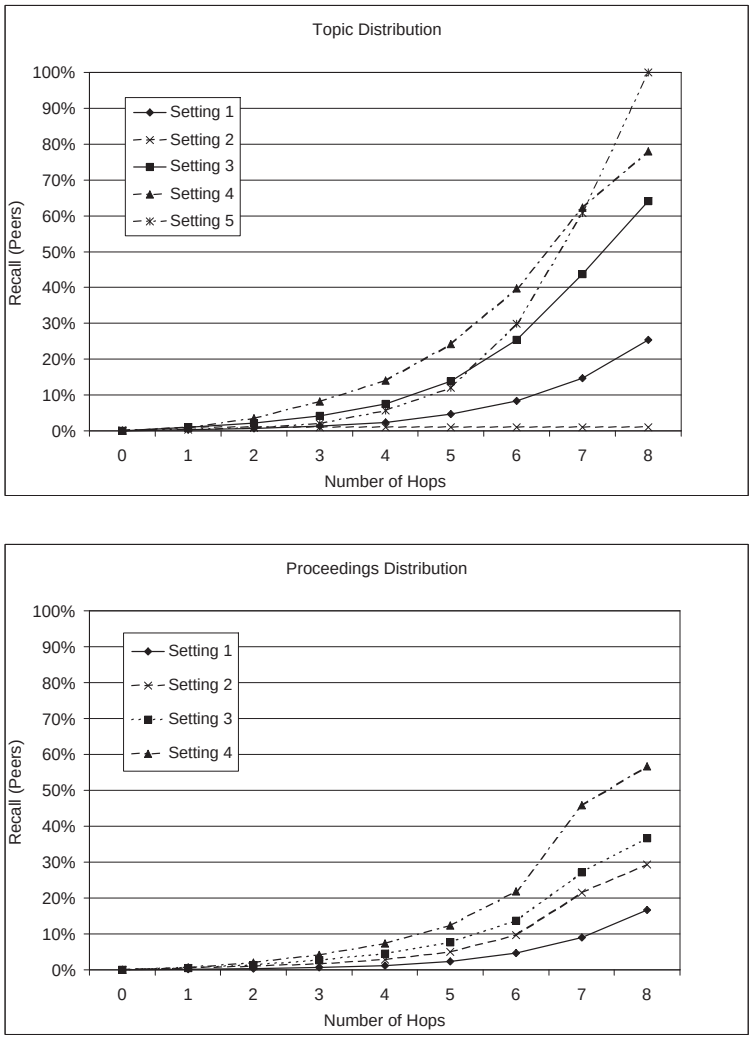


Figure 11.5: $Recall_{Peers}$ of Expertise Based Peer Selection

the peer selection, the higher is the chance of a peer receiving a query multiple times on different routes. This redundancy is detected by the receiving peer, which will forward the query only once, thus resulting in a decreasing number of queries sent across the network. The other effect is caused by the selectivity of the peer selection: It only forwards the query to peers whose expertise is semantically more or equally similar to the query than that of the own expertise. With an increasing number of hops, the semantic similarity of the expertise of the peer and the query increases, thus the chance of knowing a qualifying peer decreases, which results in a decrease of messages.

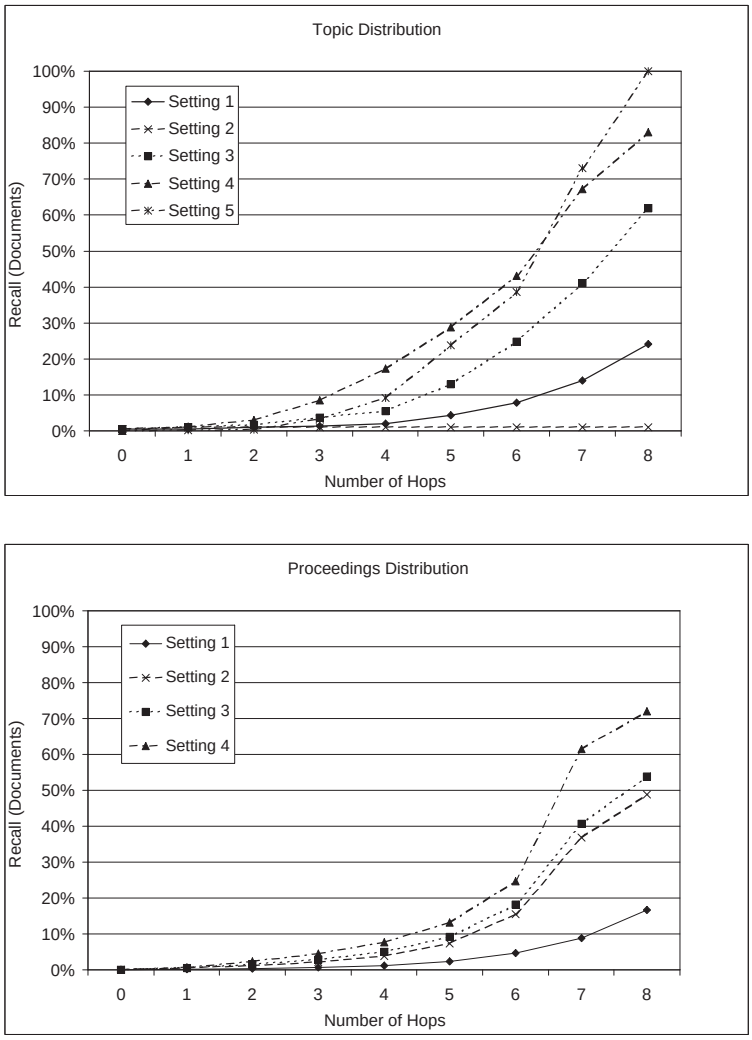


Figure 11.6: $Recall_{Documents}$ of Expertise Based Peer Selection

R2 - Ontology based matching The result of Figure 11.4, Setting 2, shows that the exact match approach results in a maximum precision already after one hop, which is obvious because it only selects peers that match exactly with the query's subject. However, Figure 11.5 shows that the recall in this case is very low in the case of the topic distribution. This can be explained as follows: For every query subject, there is only one peer that exactly matches in the entire network. In a sparse topology, the chance of knowing that relevant peer is very low. Thus the query cannot spread effectively across the network, resulting in a document recall of only 1%. In contrary, Setting 3 shows that when semantically similar peers are selected, it is possible to improve the recall of peers

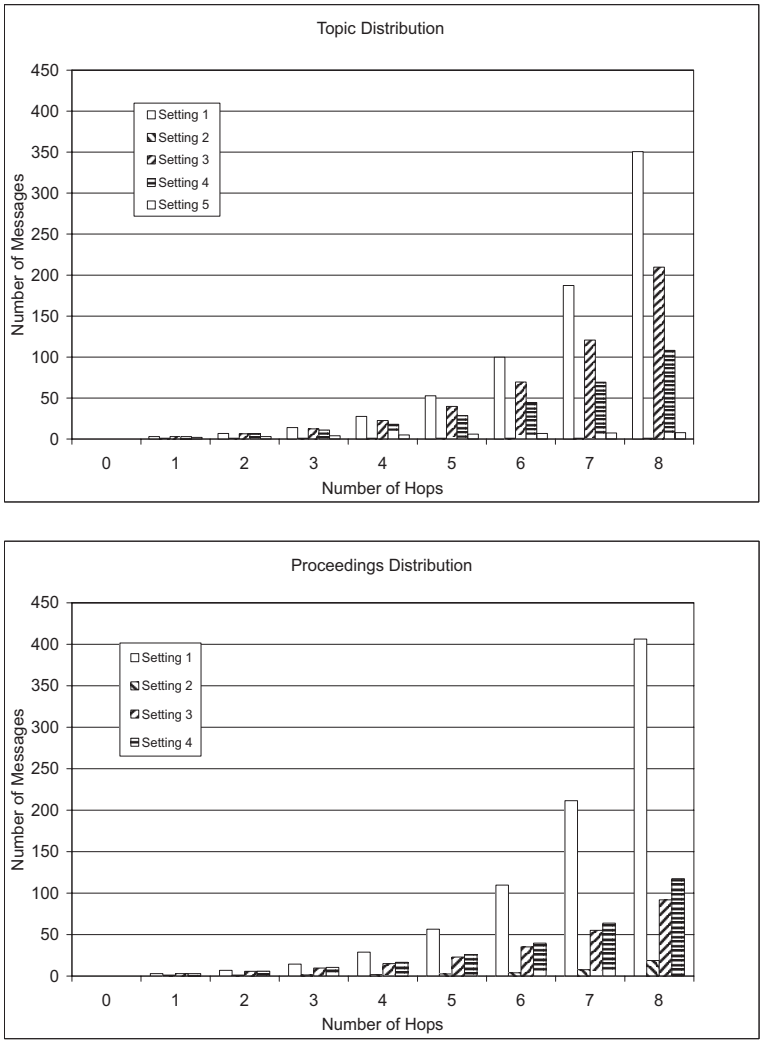


Figure 11.7: $Number_{Messages}$ of Expertise Based Peer Selection

and documents, to 62% after eight hops. Also in the case of the proceedings distribution, where multiple exact matches are possible, we see an improvement from 49% in the case of exact matches (Setting 2), to 54% in the case of ontology based matches (Setting 3). Naturally, this approach requires to send more messages per query and also results in a lower precision.

R3 - Semantic Overlay Network In Setting 4 the peers only accept semantically similar advertisements. This has proven to be a simple, but effective way for creating a semantic overlay network that correlates with the expertise of the peers. This allows to

forward queries along the gradient of increasing semantic similarity. When we compare this approach with that of Setting 3, the precision of the peer selection can be improved from 0.15% to 0.4% for the topic distribution and from 14% to 20% for the proceedings distribution. The recall of documents can thus be improved from 62% to 83% for the topic distribution and from 54% to 72% for the proceedings distribution.

It is also interesting to note that the precision of the peer selection for the similarity based matching decreases slightly after seven hops (Figure 11.4). The reason is that after seven hops the majority of the relevant peers has already been reached. Thus the chance of finding relevant peers decreases, resulting in a lower precision of the peer selection.

R4 - The "Perfect" Overlay Network The results for Setting 5 show how one could obtain the maximum recall and precision, if it were possible to impose an ideal semantic overlay network. All relevant peers and thus all bibliographic descriptions can be found in a deterministic manner, as the query is simply routed along the route which corresponds to the shortest path in the ACM topic hierarchy. At each hop the query is forwarded to exactly one peer until the relevant peer is reached. The number of messages required per query is therefore the length of the shortest path from the topic of expertise of the originating peer to that of the topic of the query subject. The precision of the peer selection increases to the maximum when arriving at the eighth hop, which is the maximum possible length of a shortest path in the ACM topic hierarchy. Accordingly, the maximum number of messages (Figure 11.7) required is also eight.

11.6 The Bibster Field Experiment

In addition to the simulation experiments, we have evaluated the methods of expertise based peer selection in a realistic field experiment, as part of the Bibster system. We have implemented the methods for expertise based peer selection in the Bibster system, and performed a public field experiment to evaluate the model in a real-world setting.

11.6.1 Setup of the Field Experiment

The Bibster system was made publicly available for download[2]and advertised to researchers in the Computer Science domain. The evaluation was based on the analysis of system activity that was automatically logged to log files on the individual Bibster clients. In Bibster two different peer selection algorithms ran at the same time, namely our expertise based peer selection and a random query forwarding algorithm. We have analyzed the results for a period of three months (June - August 2004).

[2] `http://bibster.semanticweb.org/`

398 peers spread across multiple organizations mainly from Europe and North America participated in the field experiment and used the Bibster system.

A total of 98872 bibliographic entries were shared by the 398 peers, with an average of 248 entries per peer. However, the distribution had a high variance (cf. Figure 11.8): While 62% (248 peers) were free-riding[3] and shared no content, 6% (24 peers)

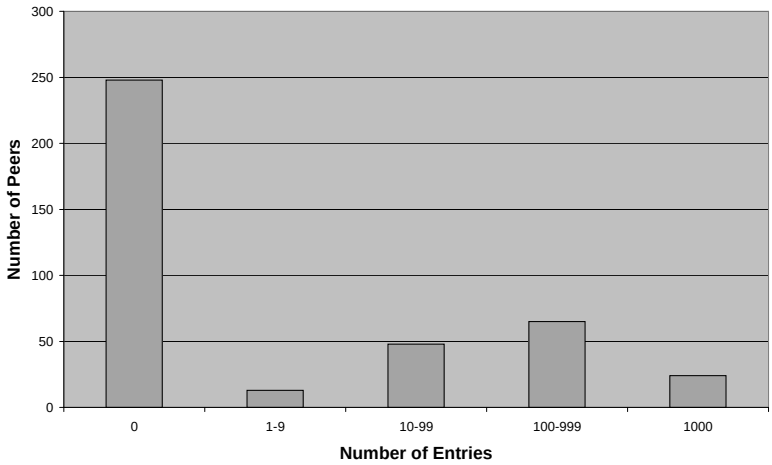


Figure 11.8: Distribution of Shared Content

shared at least 1000 entries each, accounting for 79% of the total shared content. With respect to the variance, the distribution is similar to that of the *topic distribution* from the simulation experiments, where many peers provided no entries (those whose topic had no classified instances) and few peers provided many entries (those with popular topics such as "Database Management"). The users performed a total of 3319 queries. With respect to the scope of the queries, Figure 11.9 shows that the users mainly performed queries on their local peers and automatic search across the entire network. Only in few cases the queries were directed to a manually selected peer. This confirms the need for efficient peer selection algorithms. For the 3319 queries, the users received a total of 36960 result entries, i.e. around 11 result entries per query. Result entries were actively used 801 times, i.e. copied or stored locally.

11.6.2 Results

With respect to query routing and the use of the expertise based peer selection, we were able to reduce the number of query messages by more than 50 percent, while retaining the same recall of documents compared with a naive broadcasting approach. Figure

[3]In many Peer-To-Peer systems (e.g. Naptser, Gnutella) users are mainly interested in their own advantage and conserve their resources (i.e. bandwidth) by sharing no files. In the common literature this phenomena is called *Free-Rider* problem. Users do not have a direct incentive to share files.

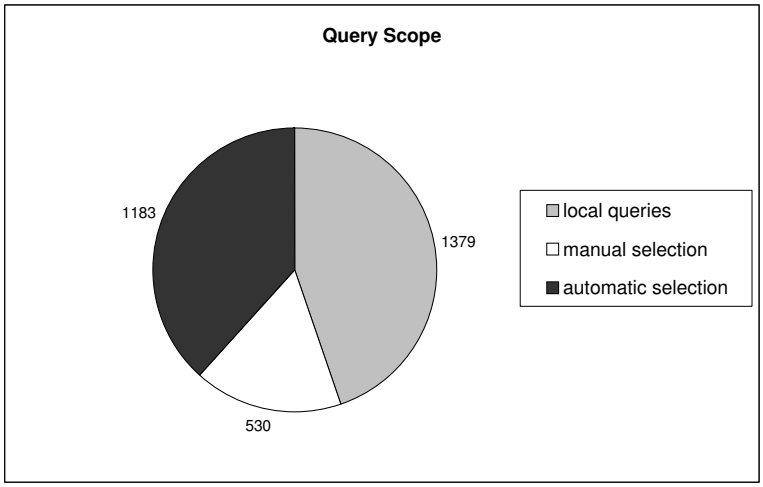


Figure 11.9: Scope of Queries

11.10 shows the precision of the peer selection (the percentage of the reached peers that actually provided answers to a given query): While the expertise based peer selection results in an almost constant high precision of 28%, the naive algorithm results in a lower precision decreasing from 22% after 1 hop to 14% after 4 hops[4].

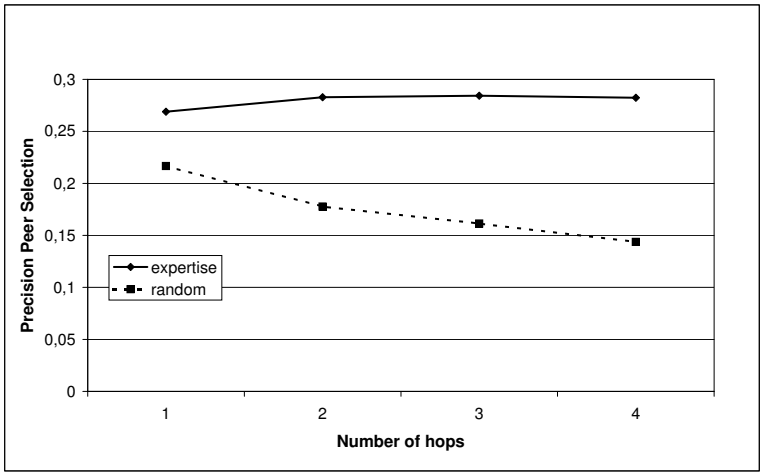


Figure 11.10: $Precision_{Peers}$ in the Field Experiment

Figure 11.11 shows the number of forwarded query messages sent per query. It can be seen that with an increasing number of hops, the number of messages sent with the

[4]The decrease is due the redundancy of relevant peers found on different message paths: Only distinct relevant peers are considered.

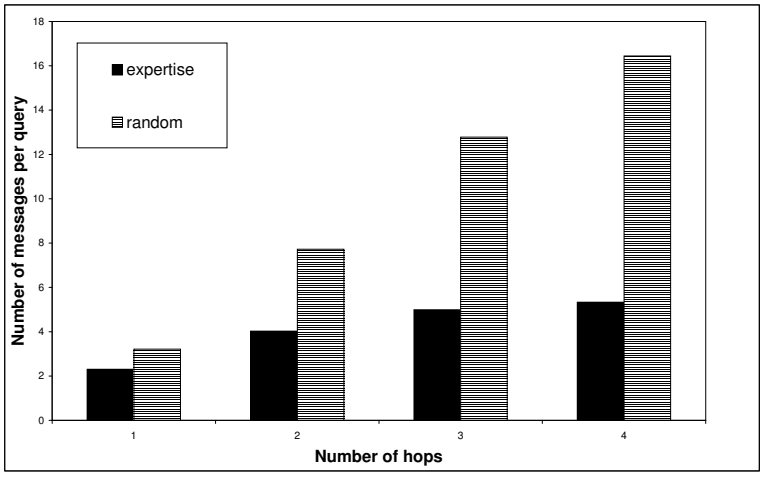


Figure 11.11: $Number_{Messages}$ in the Field Experiment

expertise based peer selection is considerably lower than with the naive algorithm. Although we have shown an improvement in the performance, the results also show that with a network of the size as in the field experiment, a naive approach is also acceptable. On the other hand, with a growing number of peers, query routing and peer selection becomes critical. In the previous discussed simulation experiments, networks with thousands of peers improve in the order of one magnitude in terms of recall of documents and relevant peers.

11.6.3 Comparison with Results from Simulation Experiments

Overall, the results of the simulation experiments have been validated: We were able to improve the precision of the peer selection and thus reduce the number of sent messages. However, the performance gain by using the expertise based peer selection was not as significant as in the simulation experiments[5]. This is mainly due to the following reasons:

- *Size of the network* The size of the network in the field experiment was considerably *smaller* than in the simulation experiments. While the total number of participating peers was already fairly large (398), the number of peers online at any point in time was fairly small (order of tens).
- *Network topology* In the field experiment we built the semantic overlay network on-top of the JXTA network topology. Again, because of the small size of the

[5]In terms of recall, there were no improvements at all, as even the naive algorithm generally was able to reach all relevant peers.

network, the JXTA topology degenerates to a fully connected graph in most cases. Obviously, for these topologies, a naive algorithm yields acceptable results.

- *Distribution of the content* In the simulation experiments, we distributed the shared content according to certain assumptions (based on topics, conferences, journals). In real-world experiments, the distribution is much more heterogeneous, both in terms of the expertise of the peers and the amount of shared content.

11.7 Related Work

In Section 11.1 we have already provided a general overview of approaches to coordination in Peer-to-Peer systems and classified the most prominent systems and technologies. In this section we analyze in more detail specific approaches that relate to semantics based coordination and show how they relate to our proposed approach.

Because of the focus of our own work on semantic overlay networks, we look closer at approaches that are based on some sort of overlay network. Freenet [CMH+02] is one approach that builds an overlay network in a completely decentralized network topology and is with this respect very similar to our proposed approach. Freenet uses a key based routing mechanism: Each node maintains a data store containing documents associated with keys, and a routing table associating nodes with records of their performance in retrieving different keys. The routing table is dynamically updated as searches and insertions of data occur. To find a document in the network given a key, the peer will first try to find it in the local data store. If this lookup fails it will forward the query to the peer with the most similar key. The same path finding process is applied to insert a document into the network: a request for the new document is made, and once it fails, the document is sent along the same path as the request. This ensures that documents are inserted into the network in the same place as requests will look for it. Initially, each peer has no information about the performance of the other peers, therefore routing of requests will be essentially random. As more documents are inserted, they will begin to cluster with data items whose keys are similar, because the same routing rules are used for all of them. Further, as data items and requests from different nodes cross paths, they will begin to share clustering information as well. The result is that the network will self organize into a distributed, clustered structure where peers tend to hold data items that are close together in key space. In difference to Freenet, where the clustering of documents is achieved by re-distributing the documents in the network (notably only based on hash keys), our approach does not re-distribute data, but instead tries to cluster peers with similar expertise, where every peer maintains local control over its data, i.e. the distribution of data is fixed. It is important to note that the similarity function used for the peer selection and placement of data simply operates on the keys, which are hash

values over the content of the data, and therefore carries no notion of semantic closeness. Further, queries are restricted to key lookups.

On the other hand, there are several approaches that try to establish more *semantic* overlay networks, based on different assumptions about the underlying network. [NWS+03] presents schema based Peer-to-Peer networks and the use of super peer based topologies for these networks, in which super peers are organized in hypercubes and peers with similar content connect to the same super peer. [LWSN03] shows how this schema based approach can be used to create Semantic Overlay Clusters in a scientific Peer-to-Peer network with a small set of metadata attributes that describe the documents in the network. In contrast, the approach in our system is completely decentralized in the sense that it does not rely on super peers.

Gridvine [ACMHP04] uses the semantic overlay for managing and mapping data and metadata schemas, on top of a physical layer consisting of a structured Peer-to-Peer overlay network, namely P-Grid, for efficient routing of messages. Here, the semantic overlay network is not used for the efficiency of query routing, but instead to realize a more semantic search than the simple key-value lookups of the underlying DHT.

Much research has focused on the problem of how to identify and describe the semantic similarity of shared content and how to build according semantic overlay networks. One approach to this is to classify the content of a peer into a shared topic vector where each element in the vector contains the relevance of that given peer for the respective topic. pSearch [TXD03], is such an example where documents in the network are organized around their vector representations (based on modern document ranking algorithms) such that the search space for a given query is organized around related documents, achieving both efficiency and accuracy. In pSearch each peer has the responsibility for a range for each element in the topic vector, e.g. $([0.2-0.4],[0.1-0.3])$. Now all expertise vectors that fall in that range are routed to that peer, meaning that, following the example vector, the expertise vector $[0.23, 0.19]$ would be routed to this peer and $[0.13, 0.19]$ not. Besides the responsibility for a vector range, a peer also knows the list of neighbors which are responsible to vector ranges close to itself. The characteristic of pSearch is that the way that peers know about close neighbors is very efficient. A disadvantage of pSearch is that all documents have to be mapped into the same (low dimensional) semantic search space and that the dimensionality on the overlay is strongly dependent of the dimensionality of the vector, with the result that each peer has to know many neighbors when the vectors have high a dimension.

Recent works as described [RGL05] propose algorithms for decentralized node clustering that identify clusters of similar nodes in Peer-to-Peer networks. Emphasis is put in particular on maintaining the node clusters in the presence of highly dynamic nodes. These approaches could very well complement our work in building and maintaining the semantic overlay network. Interestingly, the evaluation results in [RGL05] show that very good clustering results can already be obtained with simple models, e.g. where messages are forwarded only once (as in our simulation experiments, where advertisements

are only forwarded to direct neighbors). In fact, further forwarding is rather detrimental due to the higher message load.

REMINDIN' [TSW04], [Tem06] is a further interesting approach to building semantic overlay networks based on social metaphors. In contrast to our approach it does not rely on building the overlay beforehand by advertising expertise descriptions. Instead, it creates an index in a lazy manner: Peers create semantic links based on what queries other peers have asked and what answers they have provided. As the authors recognize, a combination of the lazy approach with the advertisement-based approach of our work might lead to further improvements in search efficiency. Comparing the applicability of the lazy and pro-active approaches, one can state as a general rule of thumb that lazy approaches like REMINDIN' are better suitable if index structures cannot be constructed in advance. This may be the case if the participating peers are not willing or not able to publicize their descriptions.

11.8 Conclusions

In this chapter we have presented a model for expertise based peer selection, in which a semantic overlay network among the peers is created by advertising the expertise of the peers. We have shown how the model can be applied in a bibliographic scenario. Simulation experiments that we performed with this bibliographic scenario show the following results:

- Using expertise based peer selection can increase the performance of the peer selection by an order of magnitude (result R1).
- However, if expertise based peer selection uses simple exact matching, the recall drops to unacceptable levels. It is necessary to use an ontology based similarity measure as the basis for expertise based matching (result R2).
- An advertising strategy where peers only accept advertisements that are semantically close to their own profile (i.e. that are in their semantic neighborhood) is a simple and effective way of creating a semantic overlay network. This semantic overlay network allows to forward queries along the gradient of increasing semantic similarity (result R3).
- The above results depend on how closely the semantic overlay network mirrors the structure of the ontology. All relevant performance measures reach their optimal value when the network is organized exactly according to the structure of the topology (result R4). Although this situation is idealized and it will in practice not be achievable, the experiment serves to confirm our intuitions on this.

The field experiment showed that we were able to improve the precision of the peer selection and thus reduce the number of sent messages. However, the performance gained by using the expertise based peer selection was not as significant as in the simulation experiments. Summarizing, in both the simulation experiments and the field experiments, we have shown that expertise based peer selection combined with ontology-based matching outperforms both random peer selection and selection based on exact matches, and that this performance increase grows when the semantic overlay network more closely mirrors the domain ontology.

We have made a number of simplifying assumptions in our experiments, such as the assumption that all peers agree on the use of a single ontology, which is not realistic in all cases, as we have shown for example in Chapter 9 for the use of personal ontologies. The similarity framework presented in Chapter 7 provides the required basis for similarity functions across heterogeneous ontologies. However, we expect that differences in ontologies used by different peers will *lower* our results, since the computation of the semantic distance between peers becomes less reliable across different ontologies.

In our simulation experiments, the semantic overlay network was determined once, during an initial advertising round, and was not adapted any further during the lifetime of one experiment. In our field experiment this assumption was not made and also the work in [TSW04] shows how the topology can be adjusted based on the exchange of queries and answers. The conjecture is that such a self-adjusting network will *improve* the results. We think this will be the case since the semantic overlay network will converge better towards the structure of the underlying ontology than our current one-shot advertising allows.

Chapter 12

Conclusions and Outlook

12.1 Summary

In this work we have posed the question: *How can the use of semantic technologies based on ontologies address the challenges arising from the specific characteristics of distributed information systems?*

In order to answer this question, we have started the work with an in-depth analysis of distributed information systems. We have discussed distributed information systems as systems that manage and allow to interpret data that is physically distributed, but logically related. Further we have identified three common characteristics of distributed information system: heterogeneity, dynamics, and the autonomy of nodes. These three characteristics are valid across different kinds of distributed information systems, as we have seen in the analysis of federated databases, Peer-to-Peer, and Grid systems.

We have then introduced ontologies as explicit specifications of a conceptualization. In particular, we have presented the OWL ontology language – together with extensions for rules and queries – that has provided the formal foundations for the work in the subsequent chapters. We have then shown how ontologies can be used in distributed information systems to formally specify the relationship between the data and its meaning, allowing for a clear and unambiguous interpretation of the data distributed across multiple nodes. We have further analyzed how the use of ontologies addresses the challenges arising from the characteristics heterogeneity, dynamics and autonomy of nodes:

- *Ontology-based information integration* allows to realize integrated access to heterogeneous data sources.
- *Ontology evolution* allows to manage changes in distributed information systems in a consistent manner.
- *Ontology-based coordination* allows to organize the interactions of autonomous nodes in a decentralized and to an increasing extent self-organizing manner.

The main contribution is a framework of novel semantic technologies supporting these three pillars that – put together – provide the solutions to the identified challenges and allow the realization of semantic applications.

How this can be done in practice, we have shown with the Bibster application, a semantics-based Peer-to-Peer system for the exchange of bibliographic metadata between researchers. Bibster exploits lightweight ontologies in all its crucial aspects: data-organization, query formulation, query routing, and duplicate detection. It implements many of the methods presented in this work and has thus served as a running sample scenario throughout this work.

Concluding, we now summarize the main findings this work along the three pillars *Ontology-based Information Integration*, *Ontology Evolution*, and *Ontology-based Coordination.*

Ontology-based Information Integration The first question we have addressed in this context was: *How can we express the rich semantic relationships between heterogeneous data sources described using ontologies in the absence of a single, global ontology?* As a solution, we have presented the formalization of a general mapping system for OWL DL ontologies. In this mapping system, the mappings between source and target ontology are specified as correspondences between conjunctive queries against the ontologies. The expressiveness of the mapping system is embodied in the ontology language ($\mathcal{SHOIN}(\mathbf{D})$), the supported query language (conjunctive queries), and the flexibility of assertions (GLAV approach) that do not require the use of a distinguished, global schema. We have further identified a decidable fragment of mappings and a practical query answering algorithm for the task of ontology integration. All components of the mapping system can be fully expressed in OWL DL extended with DL-safe rules. It thus integrates well with current efforts for rule extensions to OWL.

Being able to express and reason with mappings is just one aspect of integration. A second problem is that of identifying correspondences between heterogeneous sources. As this problem – as well as other integration problems – often requires the notion of similarity, we have looked at the question: *How can we formally grasp the problem of measuring similarity between heterogeneous ontologies?* We have presented a general framework for calculating similarity among elements in OWL-based ontologies. The main characteristic of the framework is its layered structure: We have defined three levels on which the similarity between two elements can be measured: data layer, ontology layer, and context layer, which cope with the data representation, ontological meaning, and contextual information about these elements, respectively. For the individual layers we have further presented a number of specific similarity measures. The measures on the data layer and ontology layer are directly defined in terms of the OWL ontology model. For the definition of similarity on the context layer we have additionally introduced the notion of contextual annotations. On the ontology layer we have defined specific mea-

sures based on standard description logic reasoning tasks, including computing concept hierarchies, instance retrieval, and realization of concepts. Finally, we have shown how similarity framework has been applied for different purposes in our motivating application scenario of the Bibster system.

Ontology Evolution As we have seen, an important aspect in information systems is the notion of consistency in order to allow useful interpretations. In the context of the dynamic characteristics of distributed information systems, we have therefore looked at the question: *How can we guarantee a useful interpretation of information by ensuring the consistent evolution of ontologies in a dynamic environment?* The solution we presented is an approach to formalize the semantics of change for the OWL ontology language (for OWL DL and sublanguages in particular), embedded in a generic process for ontology evolution. Our formalization of the semantics of change allows to define arbitrary consistency conditions – grouped in structural, logical, and user-defined consistency – and to define resolution strategies that assign resolution functions to ensure these consistency conditions are satisfied as the ontology evolves. We have shown exemplarily, how such resolution strategies can be realized for various purposes.

The existence of multiple dynamic ontologies in particular environments has lead us to the question: *Can we exploit the heterogeneity of individual nodes in a collaborative scenario to pro-actively recommend adaptations to ontologies?* Here we have presented an approach to recommend ontology change operations to a personalized ontology based on the usage information of the individual ontologies in a user community. We have adapted a collaborative filtering algorithm to determine the relevance of ontology change operations based on the similarity of the users' ontologies. The virtue of this approach lies in the fact that the characteristics of changes to multiple, distributed ontologies are exploited, such that an individual user can profit from changes other users have made. In our experimental evaluation with the Bibster system we have seen that the users actually accept recommendations of the system for the evolution of their personal ontologies. The results further show the benefit of exploiting the similarity between the users' ontologies in a personalized recommender compared with a simple, non-personalized baseline recommender.

Ontology-based Coordination We have seen that the completely distributed nature and the high degree of autonomy of individual nodes comes with new challenges for the coordination of distributed information systems. As a consequence, we have posed the question: *How can we semantically describe autonomous resources in distributed information systems in order to support coordination tasks?* As a response, we have developed a metadata ontology for semantic descriptions of resources in distributed information systems to support their interoperation and coordination. The metadata model we have described combines features of ontologies with rich metadata about the origin of

the information. The metadata ontology builds on the core of the Ontology Metadata Vocabulary OMV, and extends it with the module P-OMV to describe peers as the providers of information. To support the creation, management and use of ontology metadata in a decentralized manner throughout the ontology lifecycle, we have further presented Oyster - a registry for metadata descriptions based on OMV that allows to store, share and discover ontologies without any centralized control.

Finally, we have looked at a specific coordination task and asked: *How can we use these semantic descriptions in order to solve the coordination task of resource selection in a network of autonomous nodes?* As a solution we have presented a model for expertise based peer selection, in which a semantic overlay network among the peers is created by advertising semantic descriptions of the peers' expertise. We have shown how the model can be applied in the bibliographic scenario of the Bibster system. In simulation experiments complemented with a real-life field experiment that we performed with this bibliographic scenario, we have shown how the performance of a Peer-to-Peer system can be improved with respect to precision, recall and the number of messages. Summarizing, in both the simulation experiments and the field experiments, we have demonstrated that expertise based peer selection combined with ontology-based matching outperforms both random peer selection and selection based on exact matches, and that this performance increase grows when the semantic overlay network more closely mirrors the domain ontology.

With the implementations and evaluations we have demonstrated the practicability of our proposed semantic technologies. Building on the established OWL ontology language and related emerging standards will hopefully foster the acceptance of these technologies. Semantic technologies are continuing to mature and we expect that they will play a significant role in the development of future information systems. We hope that this work can serve as one building block in this exciting development.

12.2 Open Questions and Future Directions

While in our work we have addressed a wide variety of problems, several issues leave room for future work, others have been intentionally left out of scope of this work. Inspired by visionaries like [AAB+05], on the following remaining pages we discuss some interesting aspects as possible future research directions. We organize these according to two dimensions: Aspects related to the architecture of distributed information systems on the one hand, and aspects related to knowledge representation formalisms on the other hand.

12.2.1 Architectural Aspects

Service-oriented Architectures In the development of our methods, we have largely abstracted from architectural aspects of specific target platforms. Instead, we have focused on common characteristics of distributed information systems. A basic assumption was that the central object in such systems to be managed is that of information. However, during the last years, we have seen a move along another important dimension: that of service orientation. So called service-oriented architectures (SOAs) are beginning to gain importance. In SOAs, the central objects are services rather than information objects. Web Services are the main new paradigm for communication between nodes in distributed systems. However, the current state of the art in SOAs and Web Services lacks in the degree of use of explicit semantics and decentralization [MS03]. While much effort has been invested into languages for the semantic description if Web Services, including OWL-S [MPM+04] and WSML [dBLPF06], large scale deployments of Semantic Web Services with automated discovery, composition and execution have not yet been realized.

We have taken initial steps towards the realization of service-oriented semantics-based Peer-to-Peer systems for example with [AH04] and [HAS04], yet a true embedding of service-oriented aspects will require more work. A road map for such work can be found for example in [BBdB+05].

Model Driven Architectures A similar paradigm shift as for service oriented architectures can be observed in the context of Model Driven Architectures (MDA) [MKUW04]. The goal of MDA is to abstract from platform specific aspects and to achieve interoperability via an integrated model for data and metadata management. The Unified Modeling Language (UML) and the Meta-Object Facility (MOF) play a central role in MDA. In general, the ideas of MDA are compatible with ontology-based development in terms of goals and view of the world. For example, MDA and ontologies essentially distinguish the same metalevels of data and metadata. Briefly put, the differences are that MDA still lacks formal semantics, while ontology-based development is not as well integrated with latest advances in software modeling and engineering. As such, the technologies are in a sense complementary, and it makes sense to integrate the two, as our recent work on Ontology Definition Metamodels show [BHHS06], [BHS06]. The benefits are in both directions: MDA technologies including UML are already very well accepted and provide extensive tool support. On the other hand, ontology-based development provides formal semantics and reasoning capabilities that MDA is lacking.

Trust, Security, and Privacy The dimensions of trust, security and privacy have been left out of scope of this work. However, they do constitute important aspects of distributed information systems. With trustworthy systems we refer to systems that safely

store data, protect it from unauthorized disclosure, protect it from loss, and make it always available to authorized users.

Naturally the realization of a trustworthy system becomes more difficult with an increasing degree of decentralization and openness, as we may need to deal with faulty and unreliable data sources and participants. One dimension of trust includes techniques for ensuring the correctness of query results and data-intensive computations, which is a growing concern in many applications areas. Here, semantic technology, especially logical inferencing, may be helpful in validating correctness. Privacy is a final aspect of the broader issue of trustworthy systems we want to mention: The integration of pieces of previously unrelated information about individuals also holds significant risks as it may allow to entail information that violates the privacy rights of the individual. [AKSX02] has coined the term of a "Hippocratic Database" for a database that includes privacy as a central concern. In analogy and referring back to our definition of an information system as a system that manages facts about some domain for a specific purpose, we can imagine a "Hippocratic Information System" that does so for a *beneficial* purpose.

12.2.2 Knowledge Representation Aspects

Ontology Models We have based our work on the formal ontology model underlying the Web Ontology Language OWL. While OWL itself is a very expressive ontology language, there are certain aspects that can not be dealt within this language directly. These specifically include means to deal with networked ontologies (as opposed to single, isolated ontologies). We have discussed ontology mappings that often require an expressiveness beyond that of OWL. Besides mappings, there are other important relations in networks of ontologies, including modularization and other forms of dependencies, version spaces, etc. Supporting these networked ontologies in distributed information systems are the focus of recent research projects such as NeOn [DMS+05].

Context Awareness In many applications, the interpretation of data is context dependent, i.e. there is not one fixed purpose, but instead the task at hand, situation, location, preferences or viewpoints are part of the purpose. This is especially true for distributed settings, where ontologies and corresponding data are produced and consumed by autonomous individuals or groups in certain settings for specific purposes. Contextualization can further be supplemented by personalization, taking into the experience, interests, personal needs. Context models are currently an active research area [GMF04], [BGH+03]. As context sensitivity of applications will be a key concept for future information systems, infrastructures for semantics-based applications are expected to support contexts as first class citizens [DMS+05].

Unstructured Data In this work we have focused very much on dealing with structured data. However, much – if not most – of available data is only available in unstructured form, e.g. in the case of text, temporal, spatial, sound, image, or video data. The importance of these more sophisticated data types has increased dramatically in the last decade, and with computational resources increasing in power and decreasing in cost, their management becomes more feasible. Yet, we are far away from an intelligent management of multimedia content. In most cases, multimedia data is managed as raw data in the form of BLOBs (binary large objects). Search and retrieval is currently limited to meager structured metadata, if at all available. In the future, ontologies and semantic annotations are expected to play the key role supporting integration and fusion of information derived from different media types, be they unstructured or structured [PBS+06]. Research projects such as X-Media[1] currently investigate open problems along these lines. Based on the Bibster system, we have had first encouraging experiences in the integration of structured and unstructured textual data, which we have reported on in [HSSV05].

Natural Language Understanding Closely related to the management of text as unstructured data is the problem of natural language understanding: In order for a machine to be able to interpret natural language text and process it based on its *meaning*, we require techniques to transform textual data into more formal representations. Ontologies seem to provide the adequate formalisms, as recent results in the young research field of *ontology learning from text* indicate [CV05]. The same holds not only for understanding textual data itself, but also on the side of the user interface: It is fairly clear that end users will not learn SPARQL or any other query language for the Semantic Web that may become standardized in the future; such query languages will always be notations for professional programmers. The information-retrieval community has used keyword-based querying for decades. Recent results show that ontologies can help in answering such queries, e.g. by query disambiguation or query refinement [Sto04b]. However, what seems more promising is question answering using structured natural language queries, whose semantics is unambiguous in the first place and that additionally provide more expressiveness. We have been able to demonstrate results of first steps in these directions for example in [CHS+06].

Imperfect Information: Imprecision, Inconsistency and Uncertainty In traditional information systems, information is assumed to of "perfect" nature, meaning that it is correct and we know how to interpret it. In many application areas, such as business data processing, this assumption is useful (and often practically unavoidable). However, real-world data is very often afflicted with different forms of imperfection, e.g. if data is obtained from sensory input, or using image recognition or data mining techniques.

[1] http://www.x-media-project.org/

According to [MS97] imperfection can be due to *imprecision*, *inconsistency* or *uncertainty*. Imprecision and inconsistency are properties of the information itself - either more than one world (in the case of ambiguous, vague or approximate information) or no world (if contradictory conclusions can be derived from the information) is compatible with the given information. Uncertainty refers to a situation where we have only partial knowledge about the truth value of a given piece of information.

While in this work we have discussed methods for dealing with inconsistency in the presence of changes, dealing with imprecision and uncertainty requires different techniques. For example, in [HV05] we have reported on an approach to deal with imperfection in ontology learning, where the acquired ontologies inherently represent uncertain and possibly contradicting information. However, much more work is needed in order to incorporate other aspects of imperfection. Probabilistic and fuzzy extensions to underlying representation formalism [Str05, DP04] are possible options. Another useful concept in dealing with imperfect data is the notion of provenance: If we know where the data came from and what cleaning, rescaling, remodeling, etc. was done subsequently to arrive at the data, we may be better able to interpret it afterwards.

References

[AAB+05] S. Abiteboul, R. Agrawal, P. A. Bernstein, M. J. Carey, S. Ceri, W. B. Croft, D. J. DeWitt, M. J. Franklin, H. Garcia-Molina, D. Gawlick, J. Gray, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, M. L. Kersten, M. J. Pazzani, M. Lesk, D. Maier, J. F. Naughton, H.-J. Schek, T. K. Sellis, A. Silberschatz, M. Stonebraker, R. T. Snodgrass, J. D. Ullman, G. Weikum, J. Widom, and S. B. Zdonik. The lowell database research self-assessment. *Commununications of the ACM*, 48(5):111–118, 2005.

[Abe06] K. Aberer. Informal personal discussion at EPFL, Lausanne, February 2006.

[ACCM+04] K. Aberer, T. Catarci, P. Cudré-Mauroux, T. S. Dillon, S. Grimm, M.-S. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, A. M. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. De Santis, S. Spaccapietra, S. Staab, R. Studer, and O. De Troyer. Emergent semantics systems. In Bouzeghoub et al. [BGKS04], pages 14–43.

[ACMD+03] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-grid: a self-organizing structured p2p system. *SIGMOD Record*, 32(3):29–33, 2003.

[ACMH03] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. Start making sense: The chatty web approach for global semantic agreements. *Journal of Web Semantics*, 1(1):89–114, 2003.

[ACMHP04] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In McIlraith et al. [MPvH04], pages 107–121.

[AD98] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 254–263. ACM Press, 1998.

[ADHS05] K. Aberer, A. Datta, M. Hauswirth, and R. Schmidt. Indexing data-oriented overlay networks. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P. Larson, and B. C. Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 685–696. ACM, 2005.

[AGM85] C E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[AH02] K. Aberer and M. Hauswirth. An overview of peer-to-peer information systems. In W. Litwin and G. Lévy, editors, *Distributed Data & Structures 4, Records of the 4th International Meeting (WDAS 2002), Paris, France, March 20-23, 2002*, volume 14 of *Proceedings in Informatics*, pages 171–188. Carleton Scientific, 2002.

[AH04] S. Agarwal and P. Haase. Process-based integration of heterogeneous information sources. In *Proceedings of Semantische Technologien für Informationsportale, Informatik04*, SEP 2004.

[AKK+03] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The hyperion project: From data integration to data coordination. *SIGMOD Record*, 32(3), 2003. `http://www.acm.org/sigmod/record/issues/0309/C10.arenas.pdf`.

[AKSX02] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *VLDB*, pages 143–154. Morgan Kaufmann, 2002.

[AM99] G. Aslan and D. McLeod. Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution. *The VLDB Journal*, 8(2):120–132, 1999.

[Ari08] Aristotle. Metaphysics. In W. D. Ross, editor, *The Works of Aristotle translated into English, Volume VIII*. Oxford University Press, Oxford, 1908.

[BBdB+05] M. L. Brodiea, C. Bussler, J. de Bruijn, T. Fahringer, D. Fensel, M. Hepp, H. Lausen, D. Roman, T. Strang, H. Werthner, and M. Zaremba. Semantically enabled service-oriented architectures: A manifesto and a paradigm shift in computer science. Technical report, DERI, 2005.

[BCG05] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.

[BEE+04] P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krötzsch, L. Serafini, G. Stamou, Y. Sure, and S. Tessaris. Specification of a common framework for characterizing alignment, knowledgeweb deliverable d2.2.1v2. Technical report, Institut AIFB, Universität Karlsruhe, 2004.

[BEH+02] E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, and V. Zacharias. KAON - towards a large scale semantic web. In Kurt Bauknecht, A. Min Tjoa, and Gerald Quirchmayr, editors, *E-Commerce and Web Technologies, EC-Web 2002*, Lecture Notes in Computer Science 2455, Aix-en-Provence, France, September 2002. Springer 2002.

[BGH+03] P. Bouquet, F. Giunchiglia, F. Van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In Fensel et al. [FSM03], pages 164–179.

[BGKS04] M. Bouzeghoub, C. A. Goble, V. Kashyap, and S. Spaccapietra, editors. *Semantics for Grid Databases, First International IFIP Conference on Semantics of a Networked World: ICSNW 2004, Paris, France, June 17-19, 2004. Revised Selected Papers*, volume 3226 of *Lecture Notes in Computer Science*. Springer, 2004.

[BH04] S. Bloehdorn and A. Hotho. Text classification by boosting weak learners based on terms and concepts. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 331–334, 2004.

[BHHS06] S. Brockmans, P. Haase, P. Hitzler, and R. Studer. A metamodel and uml profile for rule-extended owl dl ontologies. In Sure and Domingue [SD06], pages 303–316.

[BHM+05] P. Brown, P. J. Haas, J. Myllymaki, H. Pirahesh, B. Reinwald, and Y. Sismanis. Toward automated large-scale information integration and discovery. In T. Härder and W. Lehner, editors, *Data Management in a Connected World*, volume 3551 of *Lecture Notes in Computer Science*, pages 161–180. Springer, 2005.

[BHS04] F. Baader, I. Horrocks, and U. Sattler. Description logics. In Staab and Studer [SS04], pages 3–28.

[BHS06] S. Brockmans, P. Haase, and H. Stuckenschmidt. Formalism-Independent Specification of Ontology Mappings - A Metamodeling Approach. In *5th*

International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2006, Montpellier, France, November 2006.

[Bis92] G. Bisson. Learning in FOL with a similarity measure. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 82–87. The AAAI Press / The MIT Press, 1992.

[BKLW99] S. Busse, R.-D. Kutsche, U. Leser, and H. Weber. Federated information systems: Concepts, terminology and architectures. Technical Report Forschungsberichte des Fachbereichs Informatik 99-9, TU Berlin, 1999.

[BKvH01] J. Broekstra, A. Kampman, and F. van Harmelen. *Semantics for the WWW.*, chapter Sesame: An Architecture for Storing and Querying RDF Data and Schema Information. MIT Press, 2001.

[BLR97] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 99–108. ACM Press, 1997.

[BLR03] A. Borgida, M. Lenzerini, and R. Rosati. Description logics for databases. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors, *Description Logic Handbook*, pages 462–484. Cambridge University Press, January 2003.

[BM04] P. V. Biron and A. Malhotra. XML schema part 2: Datatypes, W3C recommendation 28 october 2004, 2004.

[BMK+04] K. Baclawski, C. J. Matheus, M. M. Kokar, J. Letkowski, and P. A. Kogut. Towards a symptom ontology for semantic web applications. In McIlraith et al. [MPvH04], pages 650–667.

[Bor97] W. N. Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, University of Enschede, 1997.

[BR87] C. Beeri and R. Ramakrishnan. On the power of magic. In *Proceedings of the Sixth ACM Symposium on Principles of Database Systems*, pages 269–293, San Diego, CA, MAR 1987.

[Bro05] J. Broekstra. *Storage, Querying and Inferencing for Semantic Web Languages*. PhD thesis, Vrije Universiteit, Amsterdam, July 4 2005. ISBN 90-9019-236-0.

[BS02] A. Borgida and L. Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In R. Meersman and Z. Tari, editors, *CoopIS/DOA/ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 36–53. Springer, 2002.

[BVEL04] S. Brockmans, R. Volz, A. Eberhart, and P. Löffler. Visual modeling of OWL DL ontologies using UML. In McIlraith et al. [MPvH04], pages 198–213.

[BvHH+04] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. w3c recommendation. `http://www.w3.org/TR/owl-ref/`, 2004.

[CC94] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, 1994.

[CFK+01] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200, 2001.

[CFS+01] S. Camazine, N. R. Franks, J. Sneyd, E. Bonabeau, J.-L. Deneubourg, and G. Theraula. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ, USA, 2001.

[CGL01] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proceedings of the First Semantic Web Working Symposium*, pages 303–316, 2001.

[CGL02] D. Calvanese, G. De Giacomo, and M. Lenzerini. Description logics for information integration. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, volume 2408 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2002.

[CGM02] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Computer Science Department, Stanford University, 2002.

[CHS+06] P. Cimiano, P. Haase, Y. Sure, J. Völker, and Y. Wang. Question answering on top of the bt digital library. In *Posters at the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26, 2006*, MAY 2006.

[Cim04] P. Cimiano. Orakel: A natural language interface to an f-logic knowledge base. In *Natural Language Processing and Information Systems, 9th International Conference on Applications of Natural Languages to Information Systems, NLDB 2004, Salford, UK, June 23-25, 2004, Proceedings*, pages 401–406, 2004.

[CLN99] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:199–240, 1999.

[CMH+02] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communincations of the ACM*, 13(6):377–387, 1970.

[CPW01] S. Chopra, R. Parikh, and R. Wassermann. Approximate belief revision. *Logic Journal of the IGPL*, 9(6), 2001.

[CV05] P. Cimiano and J. Völker. Text2onto. In A. Montoyo, R. Muñoz, and E. Métais, editors, *NLDB*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2005.

[dBLPF06] J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel. The web service modeling language wsml: An overview. In Sure and Domingue [SD06], pages 590–604.

[Dey01] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[DFJ+04] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[DFvH+00] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. C. A. Klein, and J. Broekstra. Knowledge representation on the web. In F. Baader and U. Sattler, editors, *Description Logics*, volume 33 of *CEUR Workshop Proceedings*, pages 89–97. CEUR-WS.org, 2000.

[DK04] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[DLD+04] R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos. imap: Discovering complex mappings between database schemas. In G. Weikum, A. C. König, and S. Deßloch, editors, *SIGMOD Conference*, pages 383–394. ACM, 2004.

[DMDH02] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 662–673, New York, NY, USA, 2002. ACM Press.

[DMS+05] M. Dzbor, E. Motta, R. Studer, Y. Sure, P. Haase, A. Gómez-Pérez, R. Benjamins, and W. Waterfeld. Neon - lifecycle support for networked ontologies. In *Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies (EWIMT-2005)*, pages 451–452, London, UK, NOV 2005. IEE.

[DNB05] C. Hoong Ding, S. Nutanong, and R. Buyya. Peer-to-peer content distribution networks. In *Peer-to-Peer Computing: Evolution of a Disruptive Technology*. Idea Group Publisher, Hershey, PA, USA, 2005.

[DP04] Z. Ding and Y. Peng. A Probabilistic Extension to Ontology Language OWL. In *Proceedings of the 37th Hawaii International Conference On System Sciences (HICSS-37).*, Big Island, Hawaii, January 2004.

[DRJS03] D. De Roure, N.R. Jennings, and N.R. Shadbolt. The semantic grid: A future e-science infrastructure. In F. Berman, G. Fox, and A. J. G. Hey, editors, *Grid Computing - Making the Global Infrastructure a Reality*, pages 437–470. John Wiley and Sons Ltd., 2003.

[Ehr06] M. Ehrig. *Ontology Alignment: Bridging the Semantic Gap*. PhD thesis, University of Karlsruhe, 2006.

[EHSH05] M. Ehrig, P. Haase, N. Stojanovic, and M. Hefke. Similarity for ontologies - a comprehensive framework. In *13th European Conference on Information Systems*, MAY 2005.

[EHvH+03] M. Ehrig, P. Haase, F. van Harmelen, R. Siebes, S. Staab, H. Stuckenschmidt, R. Studer, and C. Tempich. The SWAP data and metadata model for semantics-based peer-to-peer systems. In M. Schillo, M. Klusch, J. P. Müller, and H. Tianfield, editors, *Proceedings of MATES-2003. First German Conference on Multiagent Technologies*, volume 2831 of *LNAI*, pages 144–155, Erfurt, Germany, SEP 2003. Springer.

[ESS+03] M. Ehrig, C. Schmitz, S. Staab, J. Tane, and C. Tempich. Towards evaluation of peer-to-peer-based distributed knowledge management systems. In *Proceedings of the AAAI Spring Symposium "Agent-Mediated Knowledge Management (AMKM-2003)"*, 2003.

[ESS05] M. Ehrig, S. Staab, and Y. Sure. Supervised learning of an ontology alignment process. In K.-D. Althoff, A. Dengel, R. Bergmann, M. Nick, and T. Roth-Berghofer, editors, *Wissensmanagement*, pages 487–492. DFKI, Kaiserslautern, 2005.

[Fel04] C. Fellenstein. *On Demand Computing: Technologies and Strategies*. IBM Press, 2004.

[FGM00] E. Franconi, F. Grandi, and F. Mandreoli. A semantic approach for schema evolution and versioning in object-oriented databases. In *Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, pages 1048–1062, 2000.

[FK97] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[FKMP03] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *ICDT*, volume 2572 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2003.

[FKNT02] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. Technical report, Global Grid Forum, January 2002.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 67–73, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[FPA05] G. Flouris, D. Plexousakis, and G. Antoniou. Updating DLs using the AGM theory: A preliminary study. In I. Horrocks, U. Sattler, and

F. Wolter, editors, *Description Logics*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[FPA06] G. Flouris, D. Plexousakis, and G. Antoniou. Evolving ontology evolution. In Jirí Wiedermann, Gerard Tel, Jaroslav Pokorný, Mária Bieliková, and Julius Stuller, editors, *SOFSEM*, volume 3831 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2006.

[FSM03] D. Fensel, K. P. Sycara, and J. Mylopoulos, editors. *The Semantic Web - ISWC 2003, Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003, Proceedings*, volume 2870 of *Lecture Notes in Computer Science*. Springer, 2003.

[FSvH04] C. Fluit, M. Sabou, and F. van Harmelen. Supporting user tasks through visualisation of light-weight ontologies. In Staab and Studer [SS04], pages 415–434.

[GGM+02] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with dolce. In *EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 166–181, London, UK, 2002. Springer-Verlag.

[GHVD03] B. Grossof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of WWW 2003*, Budapest, Hungary, May 2003.

[GMBM05] Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors. *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, volume 3729 of *Lecture Notes in Computer Science*. Springer, 2005.

[GMF04] R. V. Guha, R. McCool, and R. Fikes. Contexts for the semantic web. In McIlraith et al. [MPvH04], pages 32–46.

[GMK88] H. Garcia-Molina and B. Kogan. Node autonomy in distributed systems. In *Proceedings of the International Symposium on Databases in Parallel and Distributed Systems, DPDS*, pages 158–166, 1988.

[GPB02] A. Gómez-Pérez and V. R. Benjamins, editors. *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, volume 2473 of *Lecture Notes in Computer Science*. Springer, 2002.

[GPE05] A. Gómez-Pérez and J. Euzenat, editors. *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, volume 3532 of *Lecture Notes in Computer Science*. Springer, 2005.

[GPS04] B. Cuenca Grau, B. Parsia, and E. Sirin. Working with multiple ontologies on the semantic web. In McIlraith et al. [MPvH04], pages 620–634.

[Gro03] Object Management Group. Meta object facility (mof) specification, version 1.4. OMG Specification formal/02-04-03, Object Management Group, 2003.

[Gru93] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[Gru04] T. Gruber. Interview with Tom Gruber. *Bulletin of AIS Special Interest Group on Semantic Web and Information Systems (SIGSEMIS)*, 1(3), 2004.

[Gua98] N. Guarino. Formal ontology and information systems. In N. Guarino, editor, *Proceedings 1st International Conference on Formal Ont. in Inf. Sys. (FOIS)*, volume 46 of *Frontiers in AI and App.*, Trento, Italy, 1998. IOS-Press.

[Hal03] A. Y. Halevy. Data integration: A status report. In G. Weikum, H. Schöning, and E. Rahm, editors, *BTW*, volume 26 of *LNI*, pages 24–29. GI, 2003.

[HAS04] P. Haase, S. Agarwal, and Y. Sure. Service-oriented semantic peer-to-peer systems. In *Web Information Systems Engineering – Workshop Proceedings, November 22nd, 2004, Brisbane, Australia*, LNCS. Springer, 2004.

[HBE+04] P. Haase, J. Broekstra, M. Ehrig, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In McIlraith et al. [MPvH04].

[HBEV04] P. Haase, J. Broekstra, A. Eberhart, and R. Volz. A comparison of RDF query languages. In McIlraith et al. [MPvH04], pages 502–517.

[HEHS04] P. Haase, M. Ehrig, A. Hotho, and B. Schnizler. Personalized information access in a bibliographic peer-to-peer system. In *Proceedings of the AAAI Workshop on Semantic Web Personalization, 2004*, JUL 2004.

[Hen03] J. Hendler. On beyond ontology. Keynote talk, Second International Semantic Web Conference, 2003.

[HHK+05] P. Hitzler, P. Haase, M. Krötzsch, Y. Sure, and R. Studer. DLP isn't so bad after all. In *Proceedings of the Workshop OWL - Experiences and Directions, Galway, Ireland, November 2005*, NOV 2005.

[HHSTS05] P. Haase, A. Hotho, L. Schmidt-Thieme, and Y. Sure. Collaborative and usage-driven evolution of personal ontologies. In Gómez-Pérez and Euzenat [GPE05], pages 486–499.

[HJMS+00] W. Hoschek, F. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger. Data management in an international data grid project. In *Proceedings of the First International Project on Grid Computing (GRID)*, pages 77–90, 2000.

[HKTR04] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[HM78] M. Hammer and D. McLeod. On database management architecture. Technical report, Massachsetts Institute of Technology, Cambridge, Mass., 1978.

[HM05] P. Haase and B. Motik. A mapping system for the integration of owl-dl ontologies. In A. Hahn, S. Abels, and L. Haak, editors, *IHIS*, pages 9–16. ACM, 2005.

[HMS04] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In *Proceedings of the 9th Conference on Knowledge Representation and Reasoning (KR2004)*, pages 152–162. AAAI Press, June 2004.

[HMS05] U. Hustadt, B. Motik, and U. Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proceedings IJCAI 2005*, Edinburgh, UK, 2005. Morgan-Kaufmann Publisher.

[HPS04a] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL rules language. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.

[HPS04b] I. Horrocks and P. F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. *Journal of Web Semantics*, 1(4):345–357, 2004.

[HPS+05] J. Hartmann, R. Palma, Y. Sure, M. d. C. Suárez-Figueroa, P. Haase, A. Gómez-Pérez, and R. Studer. Ontology metadata vocabulary and applications. In *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, volume 3762 of *Lecture Notes in Computer Science*, pages 906–915. Springer, 2005.

[HPSvH03] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language . *Journal of Web Semantics*, 1(1), 2003.

[HS05a] P. Haase and L. Stojanovic. Consistent evolution of OWL ontologies. In Gómez-Pérez and Euzenat [GPE05], pages 182–197.

[HS05b] A. Heuer and G. Saake. *Datenbanken: Implementierungstechniken. 2. Auflage.* International Thomson Publishing, 2005.

[HS05c] Z. Huang and H. Stuckenschmidt. Reasoning with multi-version ontologies: A temporal logic approach. In Gil et al. [GMBM05], pages 398–412.

[HSB+04] P. Haase, B. Schnizler, J. Broekstra, M. Ehrig, F. van Harmelen, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. *Journal of Web Semantics*, 2(1):99–103, 2004.

[HSH+05] J. Hartmann, Y. Sure, P. Haase, R. Palma, and M. C. Surez-Figueroa. OMV – ontology metadata vocabulary. In Chris Welty, editor, *ISWC 2005 Workshop on Ontology Patterns for the Semantic Web*, NOV 2005.

[HSSV05] P. Haase, N. Stojanovic, Y. Sure, and J. Völker. Personalized information retrieval in bibster, a semantics-based bibliographic peer-to-peer system. In K. Tochtermann and H. Maurer, editors, *Proceedings of the 5th International Conference on Knowledge Management (I-KNOW 05)*, pages 104–111, Graz, Austria, JUL 2005. JUCS.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[HSvH04] P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In Bouzeghoub et al. [BGKS04], pages 108–125.

[HT02] I. Horrocks and S. Tessaris. Querying the semantic web: a formal approach. In Ian Horrocks and James Hendler, editors, *Proceedings of the*

13th International Semantic Web Conference (ISWC 2002), number 2342 in LNCS, pages 177–191. Springer-Verlag, 2002.

[HV05] P. Haase and J. Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In P. Cesar G. da Costa, K. B. Laskey, K. J. Laskey, and M. Pool, editors, *ISWC-URSW*, pages 45–55, 2005.

[HvHH+05] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies. In Gil et al. [GMBM05], pages 353–367.

[HvHtT05] Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In Kaelbling and Saffiotti [KS05], pages 254–259.

[HVS05] P. Haase, J. Völker, and Y. Sure. Management of dynamic knowledge. *Journal of Knowledge Management*, 9(5):97–107, OCT 2005.

[JEF04] J. Joseph, M. Ernest, and C. Fellenstein. Evolution of grid computing architecture and grid adoption models. *IBM Systems Journal*, 43(4):624–645, 2004.

[JLQB06] Q. Ji, W. Liu, G. Qi, and D. A. Bell. Lcs: A linguistic combination system for ontology matching. In J. Lang, F. Lin, and J. Wang, editors, *KSEM*, volume 4092 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2006.

[KAM03] A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: semantics and algorithmic issues. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 325–336. ACM Press, 2003.

[Kan99] G. Kan. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter Gnutella, pages 94–122. O'Reilly, 1999.

[KFNM04] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In McIlraith et al. [MPvH04], pages 229–243.

[Kle04] M. Klein. *Change Management for Distributed Ontologies.* PhD thesis, Free University of Amsterdam, 2004.

[KLWZ04] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[Kos00] D. Kossmann. The state of the art in distributed query processing. *ACM Computing Survey*, 32(4):422–469, 2000.

[KPSCG06] A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca-Grau. Repairing unsatisfiable concepts in OWL ontologies. In *3rd Annual European Semantic Web Conference*, Budva, Montenegro, JUN 2006. Springer.

[KS05] L. P. Kaelbling and A. Saffiotti, editors. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*. Professional Book Center, 2005.

[LBM03] Y. Li, Z. A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *Transactions on Knowledge and Data Engineering*, 15(4):871–882, July/August 2003.

[Len02] M. Lenzerini. Data integration: A theoretical perspective. In L. Popa, editor, *PODS*, pages 233–246. ACM, 2002.

[Lev66] I. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 1966.

[Lin98] D. Lin. An information-theoretic definition of similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[LWSN03] A. Löser, M. Wolpers, W. Siberski, and W. Nejdl. Efficient data store discovery in a scientific P2P network. In N. Ashish and C. Goble, editors, *Proceedings of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, CEUR WS 83, 2003.

[LYV+98] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. D. Ullman, and M. Valiveti. Capability based mediation in tsimmis. In L. M. Haas and A. Tiwary, editors, *SIGMOD Conference*, pages 564–566. ACM Press, 1998.

[Min01] N. Minar. Peer-to-peer is not always decentralized. O'Reilly Peer-to-Peer and Web Services Conference, November 2001. `http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html`.

[MKUW04] S. J. Mellor, S. Kendall, A. Uhl, and D. Weise. *MDA Distilled*. Addison-Wesley, March 2004.

[MM04] F Manola and E. Miller. Resource description framework (RDF) model and syntax specification, February 2004. `http://www.w3.org/TR/rdf-primer/`.

[MMS03a] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302, 2003.

[MMS+03b] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. An infrastructure for searching, reusing and evolving distributed ontologies. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 439–448. ACM, 2003.

[Mot06] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, University of Karlsruhe, 2006.

[MPM+04] D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. V. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. P. Sycara. Bringing semantics to web services: The owl-s approach. In J. Cardoso and A. P. Sheth, editors, *SWSWPC*, volume 3387 of *Lecture Notes in Computer Science*, pages 26–42. Springer, 2004.

[MPvH04] S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors. *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 3298 of *Lecture Notes in Computer Science*. Springer, 2004.

[MS97] A. Motro and P. Smets. *Uncertainty Management In Information Systems*. Springer, 1997.

[MS02] A. Maedche and S. Staab. Measuring similarity between ontologies. In Gómez-Pérez and Benjamins [GPB02], pages 329–350.

[MS03] A. Maedche and S. Staab. Services on the move - towards p2p-enabled semantic web services. In *Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki, Finland, 29th-31st January 2003*. Springer, 2003.

[MSR04] S. Middleton, N. Shadbolt, and D. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22:54–88, 2004.

[MSS04] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. In McIlraith et al. [MPvH04], pages 549–563.

[NK02] N. F. Noy and M. Klein. Ontology evolution: not the same as schema evolution. In *SMI technical report SMI-2002-0926*, 2002.

[NM03] N.F. Noy and M.A. Musen. The prompt suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

[Noy04] N. F. Noy. Tools for mapping and merging ontologies. In Staab and Studer [SS04], pages 365–384.

[Noy05] N. Noy. Representing classes as property values on the semantic web. Technical report, W3C, 2005. `http://www.w3.org/TR/swbp-classes-as-values/`.

[NP01] I. Niles and A. Pease. Towards a standard upper ontology. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages 2–9, New York, NY, USA, 2001. ACM Press.

[NR04] S. Nirenburg and V. Raskin. *Ontological Semantics (Language, Speech, and Communication)*. The MIT Press, September 2004.

[NWQ+02] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: A P2P networking infrastructure based on RDF. In *Proceedings to the Eleventh International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.

[NWS+03] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary, May 2003.*, 2003.

[Obe05] D. Oberle. *Semantic Management of Middleware*. Semantic Web and Beyond. Springer, 2005.

[O'R05] Tim O'Reilly. What is web 2.0 – design patterns and business models for the next generation of software. Technical report, O'Reilly, 2005. `http://www.oreillynet.com/lpt/a/6228`.

[PAP+03] E. Pitoura, S. Abiteboul, D. Pfoser, G. Samaras, and M. Vazirgiannis. Dbglobe: a service-oriented p2p system for global computing. *SIGMOD Record*, 32(3):77–82, 2003.

[PBS+06] K. Petridis, S. Bloehdorn, C. Saathoff, N. Simou, S. Dasiopoulou, V. Tzouvaras, S. Handschuh, Y. Avrithis, Yi. Kompatsiaris, and S. Staab. Knowledge representation and semantic annotation of multimedia content. *IEE Proceedings on Vision, Image and Signal Processing - Special issue on the Integration of Knowledge, Semantics and Digital Media Technology*, 153(3):255–262, JUN 2006.

[PH05] R. Palma and P. Haase. Oyster - sharing and re-using ontologies in a peer-to-peer community. In Gil et al. [GMBM05], pages 1059–1062.

[PÖ97] R. J. Peters and M. T. Özsu. An axiomatic model of dynamic schema evolution in objectbase systems. *ACM Transactions on Database Systems*, 22(1):75–114, 1997.

[PSHH04] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. w3c recommendation. `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/`, 2004.

[PSK05] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In A. Ellis and T. Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 633–640. ACM, 2005.

[PT06] P. Plessers and O. De Troyer. Resolving inconsistencies in evolving ontologies. In *3rd Annual European Semantic Web Conference*, Budva, Montenegro, JUN 2006. Springer.

[QLB06] Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *Proceedings of 10th European Conference on Logics in Artificial Intelligence (JELIA'06)*. Springer Verlag, 2006.

[RB01] E. Rahm and P. A. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Redmon, WA, 2001.

[RBJS03] D. De Roure, M.A. Baker, N.R. Jennings., and N.R. Shadbolt. The evolution of the grid. In F. Berman, G. Fox, and A.J.G. Hey, editors, *Grid computing - making the global infrastructure a reality*, pages 65–100. John Wiley and Sons Ltd, 2003.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[Res95] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montral, Qubec, Canada. Morgan Kaufmann*, pages 448–453, 1995.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM '01*, 2001.

[RGL05] L. Ramaswamy, B. Gedik, and L. Liu. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):814–829, September 2005.

[RH04] D. De Roure and J. A. Hendler. E-science: The grid and the semantic web. *IEEE Intelligent Systems*, 19(1):65–71, 2004.

[Ric92] M. M. Richter. Classification and learning of similarity measures. Technical Report SR-92-18, Fachbereich Informatik, Universität Kaiserslautern, 1992.

[RJS05] D. De Roure, N. R. Jennings, and N. R. Shadbolt. The semantic grid: Past, present and future. *Procedings of the IEEE*, 9(3):669–681, 2005.

[RMBB89] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, 1989.

[Rod95] J. F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.

[RS97] M. Tork Roth and P. M. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 266–275. Morgan Kaufmann Publishers Inc., 1997.

[SAB+05] L. G. A. Sung, N. Ahmed, R. Blanco, H. Li, M. A. Soliman, and D. Hadaller. A survey of data management in peer-to-peer systems. Technical report, School of Computer Science, University of Waterloo, 2005.

[Sal71] G. Salton. Relevance feedback and the optimization of retrieval effectiveness. In G. Salton, editor, *The SMART system — experiments in automatic document processing*, pages 324–336. Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[SASS04] L. Stojanovic, A. Abecker, N. Stojanovic, and R. Studer. On managing changes in the ontology-based e-government. In R. Meersman and Z. Tari, editors, *CoopIS/DOA/ODBASE (2)*, volume 3291 of *Lecture Notes in Computer Science*, pages 1080–1097. Springer, 2004.

[SBF98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1-2):161–197, 1998.

[SBH+05] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle. The SWRC ontology - semantic web for research communities. In Carlos Bento, Amilcar Cardoso, and Gael Dias, editors, *Proceedings of the 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005)*, volume 3803 of *LNCS*, pages 218 – 231, Covilha, Portugal, DEC 2005. Springer.

[SC03] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03*, Acapulco, Mexico, 2003. Morgan Kaufmann.

[Sch05] S. Schlobach. Diagnosing terminologies. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 670–675. AAAI Press AAAI Press / The MIT Press, 2005.

[SD06] Y. Sure and J. Domingue, editors. *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*, volume 4011 of *Lecture Notes in Computer Science*. Springer, 2006.

[SF91] P. D. Stotts and R. Furuta. Dynamic adaptation of hypertext structure. In *Hypertext'91 Proceedings, San Antonio, TX, USA*, pages 219–231. ACM, 1991.

[SGMB03] L. Serafini, F. Giunchiglia, J. Mylopoulos, and P. A. Bernstein. Local relational model: A logical formalization of database coordination. In P. Blackburn, C. Ghidini, R. M. Turner, and F. Giunchiglia, editors, *CONTEXT*, volume 2680 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2003.

[SHG03] N. Stojanovic, J. Hartmann, and J. Gonzalez. Ontomanager - a system for usage-based ontology management. In *In Proceedings of FGML Workshop. SIG of German Information Society (FGML - Fachgruppe Maschinelles Lernen GI e.V.)*, 2003.

[Sim96] E. Simon. *Distributed Information Systems: From Centralized Systems to Distributed Multimedia.* McGraw-Hill, Inc., New York, NY, USA, 1996.

[SK03] H. Stuckenschmidt and M. C. A. Klein. Integrity and change in modular ontologies. In G. Gottlob and T. Walsh, editors, *IJCAI*, pages 900–908. Morgan Kaufmann, 2003.

[SL90] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Survey*, 22(3):183–236, 1990.

[SMK+01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM SIGCOMM '01*, 2001.

[SMMS02] L. Stojanovic, A. Mädche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In Gómez-Pérez and Benjamins [GPB02], pages 285–300.

[SR03] A. P. Sheth and C. Ramakrishnan. Semantic (web) technology in action: Ontology driven information systems for search, integration and analysis. *IEEE Data Engineering Bulletin*, 26(4):40–48, 2003.

[SS02] N. Stojanovic and L. Stojanovic. Usage-oriented evolution of ontology-based knowledge management systems. In *International Conference on Ontologies, Databases and Applications of Semantics, (ODBASE 2002), Irvine, CA*, LNCS, pages 230–242, 2002.

[SS04] S. Staab and R. Studer, editors. *Handbook on Ontologies.* International Handbooks on Information Systems. Springer, 2004.

[SSDN02] M. T. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Hypercup - hypercubes, ontologies, and efficient search on peer-to-peer networks. In G. Moro and M. Koubarakis, editors, *First International Workshop on Agents and Peer-to-Peer Computing, AP2PC 2002*, volume 2530 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2002.

[SSW05] L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping language for terminological knowledge. In Kaelbling and Saffiotti [KS05], pages 576–581.

[Sto04a] L. Stojanovic. *Methods and Tools for Ontology Evolution.* PhD thesis, University of Karlsruhe, 2004.

[Sto04b] N. Stojanovic. A logic-based approach for query refinement. In *2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004), 20-24 September 2004, Beijing, China*, pages 477–480, 2004.

[Str05] U. Straccia. Towards a fuzzy description logic for the semantic web (preliminary report). In *Proceedings of the Second European Semantic Web Conference, 2005*, pages 167–181, 2005.

[SvH05] H. Stuckenschmidt and F. van Harmelen. *Information Sharing on the Semantic Web*. Advanced Information and Knowledge Processing. Springer, 2005.

[TAD+02] B. Traversat, M. Abdelaziz, M. Duigou, J.-C. Hugly, E. Poulouy, and B. Yeager. Project JXTA Virtual Network. Technical report, Sun Microsystems Inc., 2002.

[TAM+05] V. A. M. Tamma, C. Aart, T. Moyaux, S. Paurobally, B. Lithgow Smith, and M. Wooldridge. An ontological framework for dynamic coordination. In Gil et al. [GMBM05], pages 638–652.

[TDK03] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid - the grid meets the semantic web. In Fensel et al. [FSM03], pages 706–721.

[TEF+04] C. Tempich, M. Ehrig, C. Fluit, P. Haase, E. Lladó Martí, M. Plechawski, and S. Staab. Xarop: A midterm report in introducing a decentralized semantics-based knowledge sharing application. In D. Karagiannis and U. Reimer, editors, *PAKM*, volume 3336 of *Lecture Notes in Computer Science*, pages 259–270. Springer, 2004.

[Tem06] C. Tempich. *Ontology Engineering and Routing in Distributed Knowledge Management Scenarios*. PhD thesis, University of Karlsruhe, 2006.

[TIM+03] I. Tatarinov, Z. Ives, J. Madhavant, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *SIGMOD Record*, 32(3), 2003.

[TSW04] C. Tempich, S. Staab, and A. Wranik. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In S.t I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, editors, *Proceedings of the 13th international conference on World Wide Web, WWW 2004*, pages 640–649. ACM, 2004.

[Tve03] A. Tversky. *Preference, Belief, and Similarity : Selected Writings (Bradford Books)*. The MIT Press, December 2003.

[TXD03] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the ACM SIGCOMM Conference*, Karlsruhe, Germany, August 2003.

[vHSW97] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *International Journal on Human-Computer Studies*, 46(2):183–292, 1997.

[VL] H. Varian and P. Lyman. How much information? `http://www.sims.berkeley.edu/research/projects/how-much-info/`.

[Vol04] R. Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, University of Karlsruhe, 2004.

[VPZ88] J. Veijalainen and R. Popescu-Zeletin. *Standards in Information Technology and Industrial Control*, chapter Multi-Database Systems in OSI/ISO Environments, pages 83–97. North-Holland, The Netherlands, 1988.

[W3C05] *Accepted Papers of the W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington, DC, USA*, 2005. http://www.w3.org/2004/12/rules-ws/accepted.

[WB99] P. Weinstein and W. P. Birmingham. Comparing concepts in differentiated ontologies. In *Proceedings of the Twelfth Workshop on Knowledge Acquistion, Modeling and Management (KAW'99)*, Banff, Alberta, Canada, October 1999.

[WG01] C. A. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data Knowledge Engineering*, 39(1):51–74, 2001.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.

[ZKJ02] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: a fault-tolerant wide-area application infrastructure. *Computer Communication Review*, 32(1):81, 2002.

[ZSTL04] C. Ziegler, L. Schmidt-Thieme, and G. Lausen. Exploiting semantic product descriptions for recommender systems. In *Proceedings of the Second ACM SIGIR Semantic Web and Information Retrieval Workshop (SWIR '04), July 25-29, 2004, Sheffield, UK*, 2004.

In recent years we have seen a tremendous development of distributed information systems, both in terms of scale, as well as in terms of the types of distributed information systems, where classical approaches such as distributed and federated databases are complemented with Peer-to-Peer and Grid technology.

When building such distributed information systems, we are faced with three typical challenging characteristics: (1) the heterogeneity of nodes, i.e. discrepancies in the way the data is modeled and represented, (2) dynamics, i.e. changes in the structure of the system as well as changes in the information at the individual nodes, and (3) autonomy, i.e. the ability of nodes to take decisions independently, which poses additional requirements on the coordination models to achieve scalability.

In this book we show how semantic technologies – and in particular the use of ontologies – can be employed to address these three challenges: First, we illustrate how ontologies with their well-defined semantics allow to interpret the data consistently across nodes and to describe mappings for heterogeneous models and thus provide the means for information integration. Second, we show how ontology evolution allows us to deal with changes in the information in a consistent manner. Third, we present a model for ontology-based coordination to build semantic overlay networks for effective query routing and processing in the absence of centralized control. Throughout the work, we refer to the implementation of the Bibster system – a Peer-to-Peer system for sharing bibliographic metadata – to demonstrate the benefits of semantic technologies.

ISBN: 978-3-86644-100-2

www.uvka.de