The Gnowsis Semantic Desktop approach to Personal Information Management

Weaving the Personal Semantic Web

Dipl.-Ing. Leo Sauermann

German Research Center
for Artificial Intelligence
Trippstadter Straße 122
67663 Kaiserslautern

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr. Ing.)
genehmigte Dissertation

Berichterstatter:	**Prof. Dr. Andreas Dengel**
	Prof. Dr. Mehdi Jazayeri
Vorsitzender:	**Prof. Dr. Paul Müller**

Datum der wissenschaftlichen Aussprache: 5.Juni 2009

The Gnowsis Semantic Desktop approach to Personal Information Management.

Weaving the Personal Semantic Web

Leo Sauermann

Dedication

Dedicated to
the LORD
and His Son Jesus Christ
the Savior
Who changed my life in the last years
and has carved the path that I am
following
It was the LORD who gifted me with all I have and all I can
Thank you, this one is for you.

Preface

As Tim Berners-Lee, Jim Hendler, and Ora Lassila stated in their article in the Scientific American in 2001, "The Semantic Web is not a separate Web, but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." In order to do so, formal descriptions are needed, so called metadata, expressing meaning and allowing to interchange and process knowledge via computers.

An intuitive way to do so is the employment of ontologies that offer the possibility to explicitly express concepts and their relationships according to the user's preferences (for instance, there might be concepts like person or organization with a relationship works-for) based on WWW standards. Thus they define a common vocabulary and represent the user's individual mental models at the same time. However, the success of semantic technologies strongly depends on convincing as many users as possible to participate in distributed but social ontology generation.

In his thesis, Leo proposes the Semantic Desktop as an evolutionary approach and a means for personal information management. He describes a well-founded concept and an architecture combining the various bits of information from different desktop applications without changing them. Each information item is treated as a semantic web resource whether it is file (folder or document), an email constituent (*i. e.*, message, sender, recipient, attachment), an address (. . .), or a calendar entry, *etc.*. Based on that, various services are provided to generate, browse, file, annotate, or share resources just running as extensions of the operating system. The thesis discusses a colourful bunch of alternative options how theses service may be used in practice

The main part of Leo's work has been developed within the NEPOMUK project in which DFKI and 15 other partners from science and industry worked together on developing the Social Semantic Desktop. The project ended in 2008 and offers an open platform for semantic applications extending the personal desktop into a collaboration environment that supports both the personal information management and the sharing and exchange across social and organizational relations. Leo's ideas and inspiration have been a central part of NEPOMUK. Many thousands of users from science and industry already avail themselves on this open source platform for further research on semantic technologies.

Prof. Dr. Andreas Dengel
Kaiserslautern, June 2009

Acknowledgements

It is my pleasure to thank the people and institutions that made this thesis possible. First, I want to give thanks you, the Almighty God for everything received and for taking my life in your hands. You are an awesome God. I want give thanks to my wife Ingrid Brunner-Sauermann, who was with me in good and bad moments during the last years and while writing this thesis. I just love you, live on earth is beautiful with you, a PhD is nothing compared to knowing you. Many thanks go to my parents who supported me in every project I did, be it chaotic or good. My father Wolfgang Sauermann introduced me to computers and my mother Dr. Christine Sauermann helped me emotionally to stay motivated and with her expert English knowledge. Dr. Georg Sauermann, my uncle, provided various suggestions for improvement. The presence of academics in the family surely did help a lot. This work was supported by Austrian Bundesministerium für Bildung, Wissenschaft und Kultur with a Studienabschluss Stipendium (Mar 2003-Jan 2004), by the German Federal Ministry of Education, Science, Research and Technology (bmb+f), (Grant 01 IW C01, **Project EPOS**: Evolving Personal to Organizational Memories) and by the European Union IST fund (Grant FP6-027705, **Project Nepomuk**).

Special thanks go to my fellow researchers at DFKI, the best place in the world to do practical RDF research. Professor Dr. Andreas Dengel supervised this work and I am very thankful for the mentoring he has done, scientifically, personally, and in forming the text. Thomas Roth-Berghofer was leading me in this thesis, he taught me the very details how to write a thesis based on his own successfull thesis, I especially wish to thank him as he motivated me, guided me and cared for the forthcoming of this thesis.

Mehdi Jazayeri who asked about engineering and gently waited until I had an answer. Francesco Lelli for reviewing the thesis when it was growing; cutting the bad branches while watering the good ones.

I want to thank Sven Schwarz, Heiko Maus, Ludger van Elst, Michael Sintek, Malte Kiesel, Lars Zapf, Kinga Schumacher who were my direct colleagues for the last years, and all the other members of the Knowledge Management Group who supported me with good ideas and endless patience. Ludger van Elst was leading the development of the PIMO ontology within the EPOS project, he formed the formal basis for this part of the work. Sven Schwarz is responsible for the context representation at DFKI, and he influenced me since we started exchanging ideas during my diploma thesis in 2003. Gunnar Grimnes is the soul of gnowsis since 2006 and my room-mate at DFKI, taught me to take a PhD seriously not seriously. As a special credit, I wish Gunnar Aastrand Grimnes to rock as hard as 300 Spartans. The project leader of this century of man-months is Ansgar Bernardi, shephering this crowd without needing a stick.

I want to thank Mark Siebert and Pierre Smits who conducted the evaluation of the gnowsis desktop search at Siemens and have helped to keep the system installable and simple. Special thanks are directed to the students and friends who contributed their time to the Semantic Desktop while they have been at the University of Kaiserslautern

(in alphabetical order): Benjamin Adrian, Daniel Bahls, Ralf Biedert, Daniel Burkhart, Emmanuel Gasne, Dominik “Heimwege” Heim, Martin Klinkigt, Florian “Sentry” Mittag, Antoni Mylka, Günther Noak, Frank Osterfeld, Moritz Plößl.

Essential to this work are Man Luo, Dominik Heim, Benjamin Adrian, and Danish Nadeem, who wrote their *diploma theses* under my supervision and contributed to the implementation and evaluation of the gnowsis system, their research work brought us all forward.

Dominik Heim made many of the nice graphics that were used for illustrations, and the cover. He also designed the miniquire user interface and many other parts of the gnowsis user interface. Liz Turner designed the gnowsis logo. The **cover-art** was created by myself using the web service provided by **www.wordle.net**, with the introduction text of this thesis as input. Richard Cyganiak and Anja Jentzsch have accompanied the gnowsis work ever since, Anja was the first student outside DFKI who understood what this is all about, she was also the first outsider approaching us with the wish “I want to see a demo of gnowsis *now*!”. Rósa Gudjónsdóttir and Kristina “Kicki” Groth from the Royal Institute of Technology, KTH, Sweden supported me and Dominik Heim with guidelines how to evaluate HCI aspects.

I also want to thank the participants of the case studies for their time and investment.

Endless developers from other Open Source projects contributed to humanity at large by providing free software.A plethora of open source projects (see Section 7.1) was used while creating the prototype, I thank their respective authors for their work and dedication to the Open Source movement. In the same spirit I want to thank Tim Berners-Lee who made a decision that changed the world: giving away the invention of the WWW for free for us. Since 1994 director of the W3C, despite this important position and the connected responsibilities, you answer questions by newbies (and me) and still do your own code. Many professors and managers stopped caring about the technology much earlier in their career. Assuming that technology changes the world, the right people behind technology are as important.

Abstract

We should no longer ask whether we have enough information, we should rather ask if we can manage the information we have. In theory, a person could store and access all information experienced in a lifetime. But to categorize and understand this information, integrated software tools are needed. Since 2003, I have been working on the vision of a Semantic Desktop to create a tool for keeping information. Based on semantic web standards developed by the World Wide Web Consortium (W3C), the information needed by a person to do knowledge work can be integrated and organized.

In this thesis, a software architecture for the Semantic Desktop is presented consisting of ontologies, services, and applications. The *core ontology* is the *Personal Information Model* (PIMO), a personalized categorization and organisation framework. With it, a formal representation of the mental model of a user can be expressed. This personal information model is then used across applications and across domains, integrating information sources into a coherent view of the world. Existing documents are classified with multi-perspective classification, removing the limitations of hierarchical file structures.

Several *services* are defined for the Semantic Desktop architecture. They are designed to run as operating system extensions and provide functionality to store data, annotate it, and support the user in PIM activities. To provide a *user interface*, different interface metaphors are tested in various prototypes. These range from small plugins that extend existing software with Semantic Desktop functionality to complex applications that allow generic resource browsing and annotations.

The approach was *evaluated in end-user experiments* in order to find out how PIMO reflects the personal mental model of the user. It was verified to support the users in structuring their documents across applications according to their mental model and in retrieving information based on these structures. The realized implementation shows that the architecture is valid and works in real-life settings. The evaluated prototypes have shown their benefits in many person-years of productive usage.

As part of our approach to software engineering at DFKI, we released our source code as free software. Other researchers have used our prototype as a basis for their work and have provided us with valuable feedback. Within the EU research project NEPOMUK, the presented results were integrated into the popular Linux desktop KDE (version 4) and are shipped to millions of users. The Aperture framework, initiated as part of this thesis, was downloaded more than 10.000 times and is in productive use.

Contents

List of Figures

List of Tables

The room is filled with a three-dimensional constellation of hypercards, hanging weightlessly in the air. In some places, the hypercards are placed in precise geometric patterns, like atoms in a crystal. In other places, whole stacks of them are clumped together. Drifts of them have accumulated in the corners, as though Lagos tossed them away when he was finished. It is, in fact, the three-dimensional counterpart of a messy desktop, all the trash still remaining where Lagos left it.
"How many hypercards in here?"
"Ten thousand, four hundred and sixty-three", the Librarian says.
"I don't really have time to go through them," Hiro says. "Can you give me some idea of what Lagos was working on here?"
"Well, I can read back the names of all the cards if you'd like. Lagos grouped them into four broad categories: Biblical studies, Sumerian studies, neurolinguistic studies, and intel gathered on L. Bob Rife."

Neal Stephenson, "Snow Crash", 1992

I

Background

CHAPTER 1
Introduction

> When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. It can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.
>
> The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain.
> *Vannevar Bush, "As we may think", 1945* [Bus45]

Roughly since the time IBM sold their first Personal Computer in August 1981, there are systems to support people with *Personal Information Management* (PIM). But there is a gap between science and commercial software: the visionary idea of the Memex, as described above, cannot be bought as off-the-shelf product *yet*.

The need for a Memex has not disappeared. Today we store our personal records, books, and communications in e-mails on our harddisk, and every printed contract or invoice is at some point scanned for archiving. Our music collection is no longer kept on a shelve but stored on a hard drive or mobile music player. Photos that were glued into photo albums to show them to our family do not exist any more, thanks to digital cameras they are born and recorded solely in their digital form. Since the constantly decreasing cost of storage devices there is no need to delete data any more — when a disk is full, a new one is bought at the same price but with double the size (thanks to Moore's law). It seems that we will be able to keep all data we see, hear, and create during a complete lifetime, and carry it with us; as Jim Gemmell and his team have calculated in the *MyLifeBits* project [GBL+02]. Depending on the definition, the task of handling all this information can be called Personal Information Management.

> "Stuff goes in but doesn't come back out—it just builds up"
> *A participant of Boardman and Sasse 2004, p585 [BS04]*

We should no longer ask whether we have enough information, we should rather ask if we can manage the information we have. In theory, we will soon carry all our

information at our fingertips, stored in a mobile device small as a cell phone, but in practice we will not be able to reach it. At the moment, different applications employ rather simple annotation possibilities, and each application creates its own categorization scheme. File systems use a hierarchical structure of folders, as do most e-mail systems. In document management, taxonomies can be used, in a simplified version we find them as folksonomies in web 2.0 applications. But behind any categorization scheme is a user that interprets the documents, structures, and keywords. These structures are based on concepts from the real word in which he or she lives, for example the file folder "presentations" relates to the concept of giving a presentation. The folder "holiday" may contain pictures taken during a vacation, and the same word may be used as a tag in a web application, or as a category in a PIM application. But one application does not know about the categories of another, and they integrate on the basis of custom APIs and not of generic standards. These structures used for categorization can be expressed using ontologies.

The Semantic Web is an effort led by the W3C to create a framework for resource description, applicable across application and domain boundaries. This framework can be used to express ontologies. It has an extensible architecture, which allows us to use it on the desktop, coining the name "Semantic Desktop" for this combination of Semantic Web and desktop computers. Based on it, it is possible to create a standardized approach to PIM, using ontologies, connecting information across application and domain boundaries in an associative way.

In this thesis, an architecture for a Semantic Desktop is presented consisting of ontologies, services, and applications. This semantic extension of desktop operating systems fills the gap among heterogeneous applications, and allows user to express their mental model throughout these applications. Based on the technology, we then show how they improve PIM. An implementation, the *gnowsis* system, is created to test theories and evaluate the approach.

1.1. Background

gnowsis

My motive for writing this thesis was my personal quest for the "perfect way" to write down information. The *gnowsis*[1] project, my diploma thesis, defines a system that uses Semantic Web technologies on the desktop, resulting in a personal semantic wiki, personal ontologies, and an approach for integration into existing desktop applications. Started in early 2002 and published 2003, it was supervised by Gerald Reif and Mehdi Jazayeri from the Distributed Systems Lab of the Technical University of Vienna [Sau03].

[1] `http://www.gnowsis.org`

EPOS

The *EPOS*[2] project at the DFKI Knowledge Management Lab investigated a bottom-up approach from 2003 to 2005. Based on the personal knowledge workspace and intelligent assistants, information was integrated; first personally, then into an organizational memory. The results of gnowsis were integrated into EPOS. The evaluations presented in this thesis were conducted during the EPOS project, together with students and colleagues from DFKI. The software architecture developed for EPOS is also the basis for this work.

NEPOMUK

A successor project of EPOS is *NEPOMUK*[3], bringing together various European research institutions and industry partners in an Integrated Project. The vision is to create a *Social Semantic Desktop*, allowing knowledge workers to cooperate using a network of Semantic Desktops. This project started 2006 and will end 2008. In NEPOMUK we conduct detailed research on ontologies and architecture, and we use the approach in industry settings.

Related Projects

A similar goal is behind the *Haystack* project created by the Computer Science and Artificial Intelligence Laboratory of the MIT. It contains a Semantic Web user interface for the Desktop, an application that integrates all features needed by a user, such as a universal messaging client, document management, and task management. By using a semi-structured data model, it is able to capture the context of information and thus helped to elucidate the relevance of information [QHK03, Qua03]. Jack Park and the SRI Institute worked on the integrated environment *Open IRIS*, a personal cognitive assistant, part of the *CALO* project [CPG05]. Giovanni Tummarello and Chris Morbidoni developed the DBIN peer-to-peer system with similar goals, enabling groups to express shared knowledge [TMPP05]. Stefan Decker published the vision of a *Networked Semantic Desktop* in 2004 and initiated the community around this term [DF04].

Workshops

To bring together researchers working on the topic, we initiated a series of *Semantic Desktop Workshops* at the International Semantic Web Conference. The first workshop was organized by Stefan Decker, Jack Park, Dennis Quan and myself, bringing together researchers from various backgrounds and countries. Its proceedings provide a first overview on the field [DPQS05]. Based on this workshop, a community was formed and successor workshops were organized in 2006 at ISWC [DPS+06] and various related workshops and special tracks on other conferences, such as the ESWC 2006 workshop on Design for the Semantic Desktop[4].

Hands-On

Additionally a series of Hands-On programming workshops were initiated by us to nurture the developer community. Two workshops were held in 2005, then continuing annually in 2006 and 2007[5]. The focus of these events is to help deploying the technolo-

[2] `http://www3.dfki.uni-kl.de/epos`

[3] `http://nepomuk.semanticdesktop.org`

[4] `http://semdeskdesign2007.semanticdesktop.org/`

[5] For a description and links to the other workshops, see `http://www.semanticdesktop.org/xwiki/bin/view/Wiki/SemDeskHandsOn2007April`

gies and also to gather feedback from developers.

This thesis was created within a community of individuals who are roughly tied together by mutuals goals: bringing the Semantic Web alive and building a Memex.

1.2. Research on the Semantic Desktop

PIM

Several research areas form the basis of this thesis. First, Personal Information Management defines the basis of the question: *what information needs to be managed, by whom and using which tools*. The PhD Thesis by Richard Boardman [Boa04] provides a conceptual basis to PIM. Other work focusses on user interface metaphors, such as timelines used in the *Lifestreams* approach [FG96]. PIM is introduced in Section 2.1.

The problem of such approaches is that they build up the system architecture from the very ground, creating prototypes that prove the idea but open no way to adapt these technologies on a broader scale. With Semantic Web technologies, this is far easier, as the standardization of applications and data is a firm ground to build upon.

Ontologies

On a broader basis, philosophy has tackled the problem of information on a fundamental level. The important terms are introduced in Section 3.2. The field of cognitive science can help us understand how users interact with information, starting with the perspective of the user and then deducing interfaces and information models. The term *Mental Models* was discussed in [Cra43], describing how humans create inner models of the world to better understand and react to situations. *Constructivist* theory states that knowledge about the world is created by the individual, through perception and interaction with the environment. Models will be *personal* and differ depending on person, environment and interpretation. What we model is part of the questions asked by *Gestalt Theory*. Behaviour and shape of a whole is different from its parts, the distinction between objects is important, as is completeness. The idea of a *personal information model* is to represent personal (constructivistic) Mental Models in information systems.

As a formalization used in information science, the topic of *ontologies* was established. Various approaches were developed. For this thesis we concentrate on the Semantic Web and the RDF framework. For knowledge representation, meta-data languages such as OWL, RDFS or XML Topic Maps were invented. They are presented in Section 3.4. Ontologies are often used in enterprise scenarios, with focus on organizational knowledge management. At the moment, the theory and standards for the Semantic Web are set, but the adoption of these standards is rather slow. Best practices and guidelines are being established.

Semantic Desktop

We have created the *Semantic Desktop* paradigm (described in Section 4) for two main reasons: first, the technology can be used on the desktop, hopefully creating an immediate benefit for the user. Second, nearly all data found on the web or in corporate databases was created by users interacting with personal computers or other worksta-

tions. The Semantic Desktop can support the adoption of Semantic Web technologies, by allowing users to author Semantic-Web compatible data and publish it on the web. Also, data published on the Semantic Web can be reused on the Semantic Desktop.

Missing: an architecture

The main problem left open by related work is that there is no approach for an integrated Semantic Desktop. Single applications can be integrated with each other, but when multiple applications are involved, they have to be integrated in a many–to–many way. This is not feasible in practice. The alternative is to provide a dominant, "*monopolistic*" application into which all other are integrated. Related research approaches often have a software architecture that is not applicable in practice, leaving open the question of how to integrate existing applications and data or how to design the architecture in an efficient way. Our approach has to overcome these limitations.

Lack of Technology

We face two practical problems in our research. First a lack of stable semantic web technologies. To conduct research, we need stable and fast libraries that build the core technology: RDF stores, inference engines, user-interface frameworks. Ideally, these are available under open-source licenses, this way researchers can implement new algorithms based directly on existing code. In practice, we find that core technologies are often still in an experimental stage.

Lack of Formal Evaluation

Evaluating the results of Semantic Desktop research is the second problem. In the information retrieval community standardized test data sets exist, these still have to be developed for the Semantic Desktop.

1.3. Objectives

The objective of this thesis is to *create a software architecture model for the Semantic Desktop* and to *evaluate how it can improve Personal Information Management*.

The architecture model consists of three parts, which rely on each other. First is to *find an ontology for representing personal information models*. Existing ontology systems are designed with collaborative scenarios in mind, an adapted approach is needed when integrating data on the desktop.

Second, to *define new services for the desktop*, which generalize functionalities for Personal Information Management and detach these from concrete applications. We see these services as extensions to existing operating system services.

Third, to *search for user interfaces that allow users to interact with their personal information models trough the services*. Various approaches exist, of which we wanted to evaluate some on the Semantic Desktop.

By comparing this architecture with related work, we can see how cross-application PIM can be realized based on Semantic Web technology. In case studies, the effects of our approach are evaluated. We want to prove the applicability of our approach by providing an open-source functional prototype implementation.

1.4. Approach

As a prerequisite we give an overview on the fields of Semantic Desktop and Personal Information Management, including related fields such as Semantic Web. The basic approach to reach our objectives is *incremental prototyping*. Based on requirements and research ideas, a working prototype is created. This prototype is then evaluated. The knowledge gained in the creation and evaluation process is used to create a next prototype. Using an incremental approach, we can integrate new developments form the field of Semantic Web, which is quickly evolving at the moment.

For the ontologies, related work is evaluated and compared. As input OWL, RDFS, XML Topic Maps, and SKOS are used. The design rationale is to allow users to interact with it as easily as with a tagging system, but with the possibilities of formal ontologies.

A set of services needs to be implemented to provide the necessary features for PIM applications. As user interface metaphor, we implement several approaches and compare them. A promising approach is the Personal Semantic wiki.

In Information Retrieval, there is a clear methodology for research: starting with a standardized document set, the scientist develops an algorithm for retrieval and measures precision and recall values[6]. Artificial Intelligence employs a similar methodology when evaluating unsupervised learning algorithms or ontology matching[7]. For the Semantic Desktop, no such document set with a ground truth exists yet, so we evaluate our approach in end–user experiments. For evaluating, we use similar methods as Boardman [Boa04], who has chosen a design-based research paradigm, as proposed by [Car00].

The resulting system design and implementation is evaluated in different ways:

- HCI aspects are evaluated in a long-term field study using contextual interviews. In a two-month period users conduct PIM with the prototype and are interviewed about their experience. In parallel, application use is logged and measured.

- How the approach influences information retrieval is evaluated in a controlled environment with a known set of test documents.

- The Semantic Desktop architecture can be compared with the key functionalities of related work on PIM. By this, we are able to show that existing research in PIM can be realized using the proposed architecture.

[6] *Precision* defines the accuracy of calculated result values, *recall* defines the completeness. Both are values between zero and one and are common in evaluations [Rij79].

[7] For ontology matching, the OAEI initiative provides standardized test data sets. `http://oaei.ontologymatching.org`

1.5. Results

The resulting architecture consists of a set of ontologies, services, and applications. We implemented the architecture in a prototype and evaluated it.

PIMO

The *Personal Information Model* (PIMO) ontology is the integration point for the various information sources needed for PIM. Data from existing applications is represented in RDF and integrated with this model. Documents can be categorized and "tagged" with concepts. An upper ontology with common classes for PIM independent of culture or application domain was created. It provides classes such as "Person", "Topic", "Document", "Location", "Event" allowing users to answer the basic questions about information: *who, where, when, what*. The ontology can be extended by the user or with domain ontologies. The PIMO enables a user to represent a relevant part of his mental model in a formal way.

Services and Applications

Several services are defined for the Semantic Desktop architecture. They are designed to run as operating system extensions and provide functionality to store data, annotate it, and support the user in PIM. From the application side, different user interface metaphors were tried out in various prototypes. These range from small plugins that extended existing software with functionality to complex applications that allowed generic resource browsing and annotations. Most efficient were small, dedicated applications providing useful functionalities based on the services (such as the filing tool "Drop-Box").

Integrating Approaches

Compared with related work, our result replicates various approaches that were created by others, but in an integrated way. E.g. the same functionality that was created and evaluated by Richard Boardman's cross-application PIM is also available on the Semantic Desktop. The data structures needed for MyLifeBits [GBL+02] or Lifestreams [FG96] are available on the Semantic Desktop, so in theory all of these approaches can now be integrated based on the common ontologies, services, and applications.

Evaluation

The approach was evaluated in four case studies, using the aforementioned evaluation methods. The case studies included HCI experiments, long-term usage experiments, and interviews. One case study was carried out in a company context. Besides the presented evaluations, the results were also adopted by other researchers.

Viewed from the practical side, we were able to show that a Semantic Desktop was implementable and that it could be embedded with existing software architectures. We released several open source prototypes to gather feedback from other researchers and users. Within the larger NEPOMUK project, our work has provided a valuable input.

1.6. Prerequisites

As technical architecture the Semantic Web is referenced throughout this thesis, the reader is expected to have a level of familiarity with RDFS that at least corresponds to the tutorial material in [FM04, BG04].

Knowledge about XML Topic Maps is helpful because it offers a different view on knowledge representation. An excellent introduction is the *Topic Maps Handbook* by Holger Rath, Empolis [Rat03].

1.7. Cornerstone Literature

Although all cited literature is important, some articles are pillars of this work.

Finding and Reminding: File Organization from the Desktop. Barreau and Nardi have summarized their two independent studies of the ways users organize and find files on their computers [BN95].

Improving Tool Support for Personal Information Management, the Doctoral Thesis by Richard Boardman published 2004. This work was used by us as a guidance to the field of Personal Information Management [Boa04].

The Networked Semantic Desktop. The position paper of Stefan Decker and Martin Frank that sketches the path that lead to the Networked Semantic Desktop. This paper was used to build the Semantic Desktop community, many researchers cited this paper, itself it is the first paper citing our implemented prototype, gnowsis [DF04].

Designing End User Information Environments Built on Semistructured Data Models. Dennis Quans' dissertation about the Haystack platform, a novel approach for building applications based on Semantic Web technology. The structure of his thesis was a role model for my work [Qua03].

1.8. Thesis organization

This thesis is organized in four parts. The first part *"Background"*(chapters 1-4) gives background information and introduces the relevant scientific fields. The second part *"Building the Personal Semantic Web for PIM"* (chapters 5-7) describes our approach and architecture in detail. The third part *"Discussion of the Paradigm"* (chapters 8-11) shows how the Semantic Desktop architecture improves PIM in practice. The thesis ends with a summary and conclusions and by showing some open research questions. An overview is given in figure 1.1.

The chapters comprise:

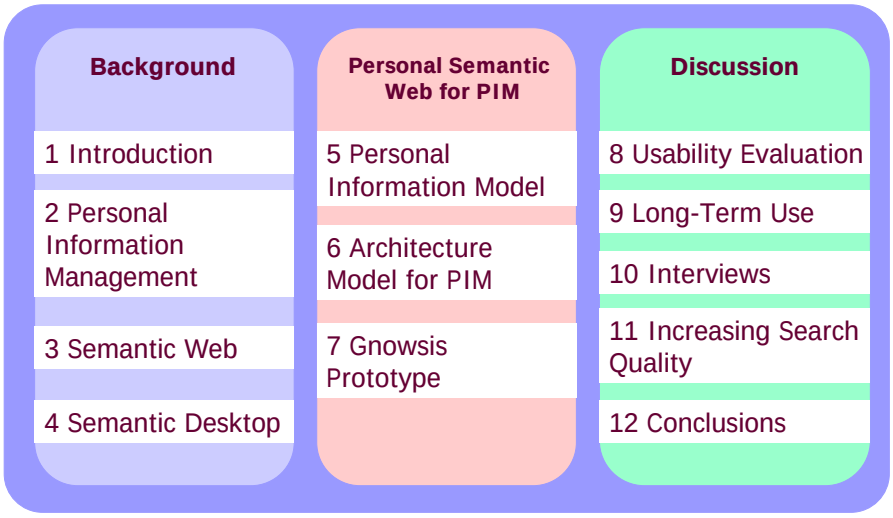


Figure 1.1.: Thesis Organization

- Chapter 2 introduces the important terms in this thesis: information, *Personal Information Management*, the Personal Knowledge Workspace. Different approaches to this area are discussed.

- Chapter 3 gives a brief introduction to the philosophical background and the cognitive aspects of information management. These philosophical topics are formalized in *ontologies*. The Semantic Web and the Resource Description Framework allow the use of ontologies on a global scale. This chapter concludes with the use of ontologies in Organizational Knowledge Management.

- Chapter 4 describes the *Semantic Desktop* paradigm. Various research threads are joined in this field, bringing together PIM and Semantic Web.

- Chapter 5 describes the data model and ontologies that we defined for the Semantic Desktop. The principal approach to data integration using *RDF* is the start, on which the *Personal Information Model* is based.

- The concept of a personal Semantic Web consisting of services, such as a centralized store on each desktop, is the key for implementing such a system. In Chapter 6 the needed services and applications are defined.

- Chapter 7 describes the implemented *gnowsis* prototype in detail. The chosen software architecture, components, inter-process communication architecture and other aspects are important to understand how a Semantic Desktop can be efficiently realized. Also this section proves that we have implemented it ourselves

based on existing technology, indicating that our approach is applicable for industrial use.

- Chapter 8 and 9 describe evaluations of the system in field studies. Long-term use of gnowsis was logged programmatically and the users were interviewed. Based on these measures, we were able to show some improvement in PIM. Finally we could indicate which parts of the software are more useful than others after two years of usage.
- Chapter 10 is a study on how the Semantic Desktop services, applications, and PIMO model match the expectations and needs of current knowledge workers. Participants were interviewed about their current PIM behaviour. After testing the Semantic Desktop, another interview was conducted about where they see the benefits of our approach.
- In Chapter 11 we describe an evaluation of the gnowsis Semantic Desktop in combination with the Brainfiler text classifier for proposal management at Siemens Business Services. The results of this study show an increase in search result quality compared to existing systems.
- The thesis ends in Chapter 12 with conclusions and outlook to future work and open problems. We also show how the implemented and realized Semantic Desktop fulfils many requirements stated by related work on PIM.

In the appendices, you find validation rules for the PIMO ontology. Also, the PIMO of the example user Paul is included. The complete source-code of our implementation is released as free software on the web[8].

[8] `http://www.gnowsis.org`

CHAPTER 2
Personal Information Management PIM

This section introduces the field of *Personal Information Management (PIM)*. The goal of this thesis is to improve it, therefore at first a definition of terms is given. Then the different tasks in PIM are described. Based on this information, tools supporting PIM are introduced.

Based on careful examination of previous designations, Boardman ([Boa04] pp. 14 and following) has created a Step-By-Step definition of the term *Personal Information Management*. It widely covers the aspects needed for the present study. However, in the course of the present investigations it appeared that additional designations were required to properly cover the extended field of interest. This necessitates to carefully describe and define the termini used in the present study, and to differentiate from those applied by Boardman and other authors.

In the following, emphasis is placed on the definition of *information*, of *personal information*, of the new terminus *personal knowledge space* and on the conception of *personal information management* PIM.

2.1. Basic Concepts

"Information" *Information* has been defined (for the context of information and library science) as "an assembly of data in a comprehensive form capable of communication and use" [FS03]. In this context information is defined more loosely as any assembly of data which carries some meaning for one or more persons. This thesis focuses on information in the digital domain: arrangements of bits which carry meaning for one or more persons, for example a paragraph of text or an image. Accordingly, this type of information is designated as *digital information*. In contrast, the definition by Shannon [Sha48] is used in computer science to mathematically describe information (and its loss) in electronic communication processes. In this thesis, not the machine's mathematical but the user's mental interpretation of information is in focus, so the definitions from library science is taken. The next stage is to distinguish *personal information* from digital information in general.

"Personal Information" *Personal information* is an ambiguous term allowing varying possible interpretations.

1. One interpretation is information *about an individual*. One common context for this usage is to describe the information stored by an institution about an individual (e.g. date of birth, credit card number).

2. A second interpretation is the information managed and stored within personal organizer software. In this sense digital personal information includes appointments, contacts, and to-do items—but not information stored outside that specific tool, such as files stored in the file system.

Boardman defines personal information as "*information owned by an individual, and under their direct control*". In other words, the owning individual is able to alter or delete the information without going through an intermediary. The current publication follows this definition. Note that this definition is independent of (1) the subject matter of the information, (2) the software application in which it is managed, and (3) the digital device on which it is stored.

Based on Boardman's definition, the problem is classifying external information that is "quasi personal information". This includes RSS feeds or reference websites. They are not stored on an individual's hard disk, but may be in the focus of tools used for Personal Information Management.

Personal Knowledge Space In addition to the content created by the individual, various information sources are accessed in knowledge work. Scientific articles created by others are stored as documents. Documentation of software systems, handbooks and tutorials, textbooks and other learning material. Private information such as e-mails from friends, photos taken by others are also collected. However, what is the difference between a scientific article stored on the hard-disk or referenced and obtainable via a hyperlink stored as a bookmark? From the perspective of the user, the information is accessible in comparable ways. The findings of Barreau and Nardi [BN95] for items stored on the hard-disk are also valid for the web.

The *Personal Knowledge Workspace (PKW)* [HMBR05] embraces all data "*needed by an individual to perform knowledge work*". It is (1) independent from the way the user accesses the data, (2) independent from the source, format, and author of the data.

Holz, Maus et al. use this approach in their paper on *Business Process-Oriented Knowledge Management* [HMBR05]. Seeing the information from the *personal knowledge space*, PIM tools can help integrate them into organizational data structures.

Boardman uses the term "personal information environment" ([Boa04], p 26), more specifically "digital personal information environment". The term "Personal Information Space" is used by [XC05]. In the key publication [JT07] the term "personal space of information"(PSI) identifies the same concept. Convening with the concepts developed in this paper the terminus *Personal Knowledge Workspace* was adopted to represent these terms.

2.2. Defining "Personal Information Management"

The Oxford English Dictionary defines "management" as "*the process of dealing with or controlling things (noun), to be in charge of an undertaking, to administer, to regulate (verb)*". Therefore, based on the above definition of personal information, and the task-oriented view of the personal knowledge space, **PIM is defined as the management of data in the personal knowledge space as performed by the owning individual**.

Boardman's definition includes the managing of information both in work and in leisure context. Related studies, such as [JMBF05] also include both work and leisure context. For our work, we concentrate on scenarios from the work context, although we understand the term "personal" to include both work and leisure.

In addition to above definitions, the following elements are presented: *Native Resources* are part of the personal knowledge workspace, including personal files of the user, e-mails, and other PIM related resources, such as appointments or contacts. In Topic Maps, the term "occurrence" is used. *Native Structures* are categorization schemes for Native Resources such as file-system folders, bookmark folders, e-mail folders, tags. In the (common) case that a user operates in a document-centered way his internal representations of these concepts are already largely reflected in content and structuring of his information elements (see, *e. g.*, [Den06b]). There are file folders called "projects", e-mail folders named after costumers' names, product names in file designators, etc. Users exploit this information for finding proper places for storing new documents, finding and re-finding old documents, or rating their relevance for current tasks. The *Mental Model* subsumes mental concepts. A mental concept is internal to the cognitive system of a person, it stands for a real-world entity or abstract concept. The inner workings of the human cognitive system are not understood thoroughly, in this thesis mental concepts are used to speak of elements of thought. Subjective to the person, the mental model is individual and cannot be externalized thoroughly.

2.3. Studies on Activities in PIM

Barreau and Nardi [Bar95] have analysed in two studies how users organize their file systems. They have based their studies on interviews with several users and by analysing their file structures. They found five functions provided by a PIM system: acquisition, organization, maintenance, retrieval and presentation of information. Users in both studies

- preferred location-based finding because of its crucial reminding function;
- avoided elaborate filing schemes;

- archived relatively little information; and
- worked with three types of information: ephemeral, working and archived

The predominant pattern for finding files was to look for a file in a particular location and then look in a different location if the file was not found. If a file was not found within a couple of tries, then a search feature was used to locate the file. The preference of location-based finding over fulltext-search on desktop computers is bound to cognition. Spatial cues are important in finding information items, as shown in [RCL+98] for the three-dimensional case and in predecessor work for two dimensions. In one interview, a user was scanning a list of files to retrieve a file she had only created that morning, saying "What did I call that file?". Despite the recency of creating the file, scanning the list was easier than remembering the name.

The remaining part of the study focused on how people bind behavioural patterns to files. For example, files kept to the right of the desktop were bound to "things to do today".

We can conclude from Barreau and Nardi's study that there are various ways to find information, and the fastest and easiest approach dominates (for example users keeping all their graphics files in the Harvard Graphics directory). Also, we see that folder structures and locations on the desktop are used with varying semantics. The three information types ephemeral, working, and archived are a categorization in the user's head, but may not necessarily be reflected in their file structures.

PIM activities

More field studies are presented in [JT07], where also a grouping of essential PIM activities in three regions was suggested (p 13, quoted literally):

- Finding/re-finding activities move from need to information. These activities affect the output of information from a PSI.
- Keeping activities move from information to need. These activities affect the input of information into a PSI.
- Meta-level activities focus on the PSI itself and on the management and organization of the PICs within. Efforts to "get organized" in a physical office, for example, are one kind of meta-level activity.

Furthermore, the same group of authors (in [Jon08] and [JT07]) provide detailed and valid insights on each activity. I was not able to use all their findings as my own implementation and field studies started in 2003. For the current study, finding and keeping activities are especially supported and analysed. In the area of finding, the problem of *re-finding* is specially addressed. Information was seen by the user before and is controlled by the user, part of the PKW. There is an aspect of re-finding information on the web, where users either keep bookmarks or navigate again to previously visited pages.

Only the bookmark and storing aspect of the web-re-finding activity is addressed by our current work.

In searching, Bonnie and Nardi observed a preference of navigating compared to full-text search. This was again observed by Teevan et al. in 2004 [TAAK04]. In their study they interviewed 15 participants twice daily on five consecutive days. They interrupting their participants' work at unspecified times and asked them to describe what they have "looked for" last. Observed usage patterns were described as *orienteering*, *teleporting*, and generally *keyword search*. Orienteering was described as navigation to the target information item using small, local steps using contextual knowledge as a guide. The conclusion was that tools should support orienteering, which (in comparison to keyword search or teleporting) lessened the cognitive burden during the search, helped to keep the sense of location, and provided users with context to understand the result and judge its value of correctness compared to the information need.

In the current study, support for orienteering is established as a side-effect based on the various semantic links established between information items.

2.4. Approaches to PIM

After having established the definition of PIM and an analysis of common tasks, software and research providing support for PIM is presented.

An approach is Eric Freeman and David Gelernter's *Lifestreams* and Gemmell's *MyLifeBits*. The assumption on which *Lifestreams* [FG96] is based is that the software systems based on the desktop metaphor are ill-equipped to manage the information of a typical computer user. They propose a new system that subsumes many desktop applications, based on the new metaphor Lifestreams, a time-ordered stream of documents. The idea is similar to Vannevar Bush's trails [Bus45]. They have built a working prototype and have described the main features of a timeline based system. In their conclusions they point out that the systems scalability can be improved, but no HCI evaluation in the sense of Barreau and Nardi was done. The *MyLifeBits* project proceeds in the same direction, but at a later stage and with more investment. A team from Microsoft Research started with the idea that all information a person will ever create and read, can be carried by that person on a mobile device. Users will be able to keep every document they read, every picture they view, all audio they hear, and a good portion of what they see. We will carry around all our information. They assume that their approach requires a hypothetical disk of one terabyte per year. Looking at the plummeting price of harddisk and better compression techniques, this assumption has not been disproven five years after publication.

Lifestreams

MyLifeBits

The key principles of MyLifeBits are:

- All relevant information a person sees, hears, reads and writes is stored in the system.
- Filesystems and directories are given up.
- Instead, an object can be assigned to zero or many collections. This allows hierarchy but does not enforce it.
- Multiple visualizations are offered to view objects. There are multiple ways to look at things, as Van Dam and Foley put it: "don't metaphor me in".
- Multimedia files are annotated with text.
- The pinnacle of value is achieved when the user constructs a "story" out of the media, a layout in time and space. This is a continuation of Trails and Lifestreams. Stories are created by transclusion, in the meaning defined by Ted Nelson. A story does not include the media, it links to it, and the links are bidirectional, building a network of information.

The aspects of visualization and a successful combination of several ideas make MyLifeBits a prototype for a useful interface.

ProFiler

Andreas Dengel et al. at *DFKI*, in cooperation with *Brainbot AG*, developed a personalized document management system called "ProFiler" [Den06b]. It has an adaptive multi-perspective user interface, consisting of multiple category trees. The main categories are addressing the aspects of classifying a document using the questions "who, what, where, and when". The categories are initially created based on existing folder structures and documents. The back-end of the system analyses the documents of a category and dynamically learns the meaning of the category, which is represented internally as term correlations. There is an option to automatically cluster documents into new categories. The user interface provides an integrated and associative management of office documents, e-mails, and bookmarks. Suggestions to classify new documents are given by analysing any new document and comparing it to existing categories. For document retrieval, the interface allows a combination of full-text, metadata, and category search. The system records successful query processes for reuse and further allows users to exchange information within a work-group. The back-end of ProFiler was used in this thesis for the gnowsis v0.8 prototype.

Microsoft Outlook

As last input for PIM tools, I want to point to a commercially successful product: *Microsoft Outlook* . This is a tool that is bound very tightly to the term "Personal Information Management". It can handle data needed in daily work situations and provides:

- A ToDo-List,
- A calendar with events,

- An integrated e-mail client,
- A timeline and planning support,
- Features to organize events and invite people,
- File management.

It is omnipresent on Microsoft Windows computers, and many applications are compatible with it enabling information exchange with it and amongst each other. One reason for this compatibility is the powerful API of Outlook and the various possibilities to adapt and enhance it. The database is extensible allowing new applications on top, or enhancement of existing applications with more data. This facilitates data integration: *as long as users have MS-Outlook, their data can be integrated*.

2.5. Cross-Tool support for PIM

Richard Boardman created the WorkspaceMirror (WM) system to evaluate the effects of cross-tool support on PIM. He recognized that users tend to replicate folder structures in multiple tools. Often the same folder names and structures created for e-mails also occur in the file-system and in bookmarks. A thorough analysis of PIM led to the design principle of *folder mirroring*. Changes of folder structure in one tool are also replicated in other tools. Similarly the events of creating, deleting, and moving/renaming folders are mirrored. In a long-term study most participants indicated that folder-mirroring was appropriate for top-level folders only. Users weighed the organizational flexibility of different folder structures against the consistency offered by WM. Design rationale for other systems was suggested, but most important is the detailed description of the evaluation.

We see that folder-mirroring is a good approach of bringing a homogeneous categorization scheme to multiple applications. The results of the evaluation are interesting. However the approach of WM may also be criticized: folder-mirroring only fights the symptoms innate to folder structures, evidently the *underlying problem is the existence of multiple categorization schemes in the first place*.

These considerations are addressed in our own approach, which evolved in parallel to WM. In this thesis, the Personal Information Model is proposed as a unifying categorization scheme, individual for the user and not for the application.

2.6. Desktop Search Engines

Desktop search applications are not new to the industry. Only the high interest in this area is new. For example, applications such as Enfish Personal[1] have been available since 1998, usually under a commercial license. As the amount of searchable Desktop data has reached very high values and will most probably also increase in the future, the major search engines have begun to put a lot of effort into this area as well. Thus, several Desktop search distributions have been released for free (e.g., Google Desktop Search[2], MSN Desktop Search[3], etc.). Moreover, some providers even integrated their Desktop search tool into their operating system, such as Apple[4]. Finally, the open source community has also manifested its interest in this area, the most prominent approaches being those of Gnome Beagle[5] (now also integrated into SuSE) and KDE KAT[6], developed within the Mandriva community. Many other commercial Desktop search applications exist (e.g. by Copernic, Yahoo! Desktop Search, X1, Scopeware Vision, PC Data Finder). However, in this Section main interest is placed on the exploitation of data sources and on the extraction and use of metadata. Therefore the particularities of the various tools are not further pursued.

DynaQ

A scientific desktop search engine is the *DynaQ* [ARD06] project. Like traditional desktop search engines, it supports keyword and metadata search. Beyond that, it supports boolean operators (including NOT), range queries, fuzzy and phonetic search, weighting of search terms, and similarity search. To make search interactive, the search results are immediately updated when the search parameters are changed, allowing the user to explore the personal space of information interactively.

Most of the above mentioned applications target an exhaustive list of indexed file types (ppt, doc, jpg, txt, html, pdf, etc.), including many basic metadata (author, title, date, size, etc.) associated with them. Also, they update their index on the fly, thus inherently tracking any kind of user activity in a certain context. However, all of them seem to only rely on the basic information provided by the data sources, without generating any *semantic information* about the objects stored on the Desktop. Thus, they obviously miss the contextual information often resulting or inferable from explicit user actions (tagging) or additional background knowledge. Teevan et al. criticise this limitation of search engines in their well-known paper "The perfect search engine is not enough: a study of orienteering behavior in directed search"[TAAK04]. In the Semantic Desktop work, we want to make use of semantic relations to improve the user experience in

[1] http://www.enfish.com/
[2] http://desktop.google.com/
[3] http://toolbar.msn.com/
[4] http://www.apple.com/macosx/features/spotlight/
[5] http://www.gnome.org/projects/beagle/
[6] http://kat.mandriva.com/

information retrieval.

2.7. A User's Work Context

What is the user currently doing and what are the goals he has in mind? Knowing the answer to this question can determine how applications present information and what options are offered. An answer to this question is sought in the field of *the user's work context*. For this work, the definition by [SRB03] is used: by observing what a user has accessed applications can infer on what problem the user works and how the user is solving this problem. Based on this information, applications can adapt to the user.

When people use computers to write down information, this information is never new. It is always created in a certain context, the individual and common background. As it is a mixture of existing information and a few new ideas, the Semantic Desktop should provide an environment where users can express new ideas and easily (preferably automatically) connect it to both personal concepts and common ontologies. We can call the background information that lead to the creation of the information resource X the *context* of the resource X. Respecting the *context of a resource* is a key feature of the Semantic Desktop. What the user is doing, what the user was doing in the last hour, day, year; what topics are relevant to the peers and the company of the user; this and much more may be used to capture this context.

We also see that the *context may switch*: while most of a user's work is around topic X (for example a project) there may be a certain time during the day (for example around noon) when the user switches to another context Y (that may be: what am I going to eat?). These context switches have to be detected and can be used. The goal of this proactive, context-sensitive assistance is that the user can keep on working as usual and the machine observes the actions of the user, automatically clustering and structuring the information at hand. Another aspect is, that the context capturing and context use is application independent. The problem Tim Berners-Lee describes should now be solved: "I saw one protagonist after the next shot down in flames by indignant researchers because the developers were forcing them to reorganize their work to fit the system" [Lee00].

Let us first look at the term "context", a part of the "*user's work context*" topic. The paper "Understanding Context Before Using It" [BB05] motivates us to look closer before taking the next steps. The term context is used in linguistics, psychology, artificial intelligence, economy, etc. Bazire and Brézillon analysed 150 different definitions of context, first by using LSA (Latent Semantic Analysis) and then STONE (a question-answer system to build a taxonomy). They come to this conclusion:

> Our conclusions of the corpus study lead us to build a model of context representing the components of a situation (where the context is taken into

> account) and the different relations between those components, with the hypothesis that the reason why definitions diverge is that they don't put their focus of attention on the same topics.
> Then, the topic on which definitions put the focus on may allow discriminating definitions. A situation could be defined by a user, an item in a particular environment, and eventually an observer (according to certain definitions). Context interferes with each of these elements (...)
> We separate context and environment because we consider that the physical environment is not totally relevant in a task running and we propose that context represents all that it is significant at a given moment but can belong to any of the terms of our model, depending to the task goal.

They come to a definition of context that covers multiple domains:

> The context acts like a set of constraints that influence the behaviour of a system (a user or a computer) embedded in a given task.

Based on this, we can now take a close look on the work context of a user. What kinds of topics is a user working on? Are these topics organized in projects and tasks? Can this work context be measured to support the user with information related to the current work?

In [Koy01] an algorithm is described for understanding drifting and recurring user interests based on machine learning. The algorithm uses a meta-learning level for learning the current context, searches into the past observations for episodes that are relevant to the current context, "remembers" them and "forgets" the irrelevant ones. Finally the algorithm learns only from the selected relevant observation. The experiments conducted with a data set about calendar scheduling recommendations show that the presented algorithm improves the predictive accuracy significantly.

Schwarz has continued his work about user observation and work context elicitation, in the EPOS project in which also gnowsis was developed. Multiple components observe the user and report activities to a central component, collecting the observations, called user observation hub. The context in which the user is doing knowledge work can then be modelled based on different aspects:

- The Organisational aspect captures the role of the user.
- Operational aspect contains active applications and used services.
- Informational aspect contains touched objects and relevant domains.
- Causal aspect contains task concepts or workflow instances.

- Behavioural aspect contains the exact actions the user did (moving the mouse, etc.).

- Attentional aspect contains the scope of read text.

- Historical aspect contains previous tasks and actions.

- Environmental aspect contains the current room, present persons, or devices.

Each element is represented using *contextual elements*, a contextual element containing the aspect in question and how relevant they currently are. Different algorithms can be used to analyse the native operations (behavioural aspect) of the user and transform them to contextual elements. Schwarz implemented a Bayesian-like algorithm of multiple contextual elements supporting each other. These results are published in [Sch06].

The L3S group published a paper on "Activity Based Metadata for Semantic Desktop Search" [CGG+04] describing a detailed ontology to represent the contextual information about several user activities, tested in a prototypical implementation. Relevant to context are e-mails and the way attachments are handled, the file hierarchy and how it resembles the users view of the world and the web browsing behaviour of users. They propose an architecture to capture these contextual elements by metadata generators. The benefit for the user is that the context is used to enrich search results in desktop search. A practical implementation of this and other ideas is shown in the *Beagle++* prototype.

Another approach evaluated at the DFKI in the eFisk project [MAD05],[MA05] is to capture the reading behaviour of the user with an eye-tracker. Using this technology, it is possible to capture on which parts of the screen the user is looking for how long. Combined with the currently displayed text, the system can recognize that the user looked a certain amount of time at a certain text. So we can assume that the text has been read and set metadata to value this text higher – during searching, we can rank read passages higher. This adds more information to the personal mental model of the user.

The current *user work context* gives PIM applications background information about the user and the task at hand. Knowing it can be used to adapt the applications. There are more projects aiming at capturing context information and representing it. We expect to see a common ontology for context information in the next years, that could connect these different approaches.

2.8. Summary on Personal Information Management

We know that users pick multiple ways to view information, categorize it and search for it, and we know that the different user interfaces to personal information offer categorization and search facilities. But we miss one aspect in related work, namely what meaning the categories did have when created by the user.

The motivation and thoughts that lead to categorization schemes are part of the mental models and cognition of the user. Based on the related studies, we see that the problem has been identified, but that the offered solutions can fix the symptoms (make the user find files faster) without solving an underlying problem: the decisions taken during PIM activities (independent of task, information type, and tool) are based on mental models of the user.

In the previous section we have seen how personal information management applications realize various categorization schemes to present data to the user. To improve PIM further, and to connect these various approaches, improved data structures are needed that address the level of mental models. Capable data structures can be found in different research threads, namely the web and the semantic web.

CHAPTER 3
Information Management on the Web and the Semantic Web

The World Wide Web ("web", "WWW") is a quite recent technical system. At the time of writing, it is roughly 15 years old, and did fundamentally change the way our society exchanges information. A main capability provided by the web is to publish documents. Hypertext documents and multimedia documents can link to other documents, using URIs for references, HTML as encoding scheme, and HTTP as a protocol for retrieval. The web as technology has an underlying base, the design of systems for organizing information rests on an intellectual foundation [Sve00]. In this chapter first the technical basis of the web is presented, then the philosophical background leading to the Semantic Web.

3.1. Hypertext and Hypermedia

In 1945, Vannevar Bush wrote the now famous article "As we may think" [Bus45], where he described a visionary system called "Memex". This vision predicted (quite accurately) the path taken by many following systems:

> Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, "memex" will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

Bush founded his ideas on the technology available after the war: analog devices, programmed with punch cards, using microfilm as storage. Today we notice how his vision becomes reality with silicium technology. The personal computer is very close to what Bush had in mind. The memex allowed keeping various media in one storage. It was envisioned as a table. To navigate, the files could be viewed on a two-monitor system on top of the table. Navigation paths can be recorded, these *trails* can be used by a person to make collections of connected files, for example a personal literature collection about a topic. The trails, including the documents, could be transferred to other memex users

via electric communication (telegraphy). One document can be part of multiple trails. The idea of *trails* reappears in systems like lifestreams [FG96]. Lifestreams uses the time order of documents as an organizational metaphor for its storage system. Stream filters are used to organize, monitor and summarize information for the user. Combined, they provide a system that subsumes many desktop applications. Still, there is work left to create the *intimate supplement to memory*. In 1960, Ted Nelson described a system called *Xanadu* in his article "*As We Will Think*" [Nel72]. Xanadu is a predecessor of hyperlink systems, the core idea was to link information items and, in a second phase, make them tradeable as a basis of information society. Nelson also coined the term "Hypertext". Although different implementations and prototypes of Xanadu were built, it never ignited the revolution that was intended by Nelson.

Before the web lifted off, Berners-Lee programmed the *Enquire-Within-Upon-Everything* system at CERN [Lee00, BLP80]. *Enquire* was a personal information management tool to store information about people, projects, hardware resources and how they relate to each other. Both Enquire and the web have been created out of a certain need:

> What I was looking for fell under the general category of documentation systems – software that allows documents to be stored and later retrieved. This was a dubious arena, however. I had seen numerous developers arrive at CERN to tout systems that "helped" people organize information. They'd say, "To use this system all you have to do is divide all your documents into four categories" or "You just have to save your data as a Word Wonderful document" or whatever. I saw one protagonist after the next shot down in flames by indignant researchers because the developers were forcing them to reorganize their work to fit the system.[Lee00, p. 17]

In 1992 the World Wide Web launched. The core standards underlying the technology and architecture are (the first two are developments of the internet and existed before):

- The TCP/IP network protocol provides the underlying communication system.
- The Domain Name System (DNS) provides a global system to identify servers.
- Unique Resource Identifiers [BLFM05] to identify resources on the web. They consist of protocol, server, and path on the server.
- The Hypertext Markup Language (HTML) to author hypertext documents. Documents can link to other documents by URI.
- The Hypertext Transfer Protocol (HTTP) to retrieve documents from servers.

The three pillars URI, HTML, HTTP provided by Tim Berners-Lee are strikingly simple to understand and to implement. There have been many other systems before and in parallel to the web (gopher, telnet, fidonet) with similar goals, but the web technologies provide the needed functionality with a minimal effort. Also, Tim Berners-Lee was aware that he cannot make the technology proprietary or commercial, otherwise it would not lift off [BL], hence the standard was published free and open. The Web changed society in a way probably only comparable to the invention of the printing press by Gutenberg. Gutenberg's movable types started in 1439[1] and then spread throughout Europe and the World in the next 50 years. After inventing the web in 1989 it was a global phenomenon within 10 years, and after 15 years the blogging systems "movable type" and "wordpress" allow every person connected to the web to become a publisher on a global scale (with a diminishing financial investment of about 3$ per hour in an internet cafe) — a situation unthinkable and unprecedented in human history. When we look at the changes movable types have caused to European politics and fostering renaissance, we can expect that the web-era will also cause changes in society structures, if it has not done so already. Tim Berners-Lee answers to the question of what he had in mind when he created the web:

> The dream behind the Web is of a common information space in which we communicate by sharing information. Its universality is essential: the fact that a hypertext link can point to anything, be it personal, local or global, be it draft or highly polished. There was a second part of the dream, too, dependent on the Web being so generally used that it became a realistic mirror (or in fact the primary embodiment) of the ways in which we work and play and socialize. That was that once the state of our interactions was on line, we could then use computers to help us analyse it, make sense of what we are doing, where we individually fit in, and how we can better work together [BL].

The interesting fact is, that the Web had its revolution in the distributed world but the topic of personal information management remained the same. The field of "documentation systems" and "Desktop Operating Systems" is still a vivid arena with many competing companies. Hypertext has not yet become the main interface for desktop computing. In this thesis, we want to present ideas that may support desktop computers with the benefits of hypertext, and create a small personal web for each user.

3.1.1. Weblogs

A weblog, or blog, is a diary that is published on a website. The action of entering new content is called "blogging". A "blogger" is a person who keeps such a weblog.

[1] `http://en.wikipedia.org/wiki/Spread_of_printing`

On February 18, 2002, *wired* magazine wrote that blogging crossed the tipping point from a "self contained community" to a major movement [Man02]. In 2007, blogging has truly become a major movement and bloggers have positioned themselves as part of contemporary media. There is a multitude of weblogs and there are more to come. The main feature of a weblog is that the blogger is able to publish a message by entering a text and pressing a button. This simplicity is striking. The weblogs have two major purposes: publishing ideas and keeping an archive. Weblogs contain many links to previous weblog entries by the same author or related weblogs by other authors. Often a weblog entry discusses a website, then there is a link to the website. Searching in a weblog can be done by date or keyword. An interesting fact is that bloggers often use their own weblog to search for information they have used some time ago. As the weblog is always online, a blogger can use it as an online information management tool. Weblogs can be integrated into the Semantic Web, as Steve Cayzer has outlined in a paper. His focus was on publishing existing metadata [Cay04].

3.1.2. Wikis

A wiki is a web content management software that allows users to enter information in a simpler syntax than html. The key feature of a wiki is that it creates hyperlinks to other pages in the wiki system automatically. There are different approaches, the most common is that words with two capital letters, like DiplomaThesis, are automatically hyperlinked to the page that has the name DiplomaThesis. These keywords are called "Wiki Words" for their use in a wiki or "Bumpy Words" or "Camel Case" because of the two upper-case letters or "bumps". Web pages in a wiki are written as plain text files through a web interface, the text can contain special formatting characters and Wiki Words. Every page has a unique name, a Wiki Word identifying it. If a Wiki Word is entered in a page and no corresponding page with the name exists, a new page can be created automatically by clicking on the word or a special tag near to it. Most public wikis are open for everyone, every user can change all content. Wiki systems are extensively used for collaborative documentation of large systems. Wikis are also a good tool for information management, some offer bidirectional links and basic topic management capabilities (using categories or tags).

We can summarize the wiki structures as following:

- Each page is identified by one unique *wiki name*. The name is a string.
- Each page is described using one longer string of *wiki markup*.
- Within the markup, one page can refer to other pages using their wiki name.
- Hyperlinks can be used to refer to other documents from the web by their *URI*.

Hundreds of different implementations exist in all major programming languages, some applications have modified the original wiki guidelines.

Nowadays, wikis are used for a wide range of applications, from the well–known Wikipedia [2], to corporate intranet applications, and personal wikis that are the equivalent of a personal notepad.

Finally, the merger of a wiki and a weblog is called a *"bloki"*. These systems provide both a way for writing blog items as short notes that are retrievable via their creation date, and have articles that are valid throughout the whole system, and throughout time, the wiki pages. With these abilities, blokis combine the benefits of both systems.

3.1.3. Semantic Wikis

Several wiki implementations exist that implement the basic wiki features and also want to address the problems indicated above. In [Ore05], an overview of semantic wikis and personal wikis is given, resulting in the description of *SemperWiki*, addressing the problems of Semantic Desktop wikis. Lets take a look at the ways metadata is implemented in different wiki implementation in the following.

In most traditional wikis, the idea of metadata typically only appears in a very technical way. For example, in *JSPWiki*[3], metadata is added directly into the wiki text using special tags, and mostly serves the purpose of implementing access control.

In *SnipSnap*[4], metadata comes by ways of labels that can be attached to wiki pages which are a kind of categorization scheme.

The semantic wiki *Platypus*[5] adds RDF(S) and OWL metadata to wiki pages. Metadata has to be entered separately from wiki text and relates a wiki page to another resource; thus, metadata can be transformed into a list of *related pages* that can be shown along with the actual wiki page.

The *Semantic MediaWiki*[6] [KVV05] is an extension of *MediaWiki*[7], the software used by Wikipedia. Again, metadata associated to a wiki page may point to other resources, but here, also data literals are allowed. Also, metadata is entered directly into the wiki text, and does not have to adhere to a schema.

Rhizome[8] builds on a framework that adapts techniques such as XSLT and XUpdate to RDF. In essence, RDF is used throughout the framework for almost everything, and RxSLT (an XSLT variant adapted for RDF) is used for transforming queries' results to

[2] Wikipedia, the free encyclopaedia. `http://wikipedia.org/`
[3] http://www.jspwiki.org/
[4] http://snipsnap.org/
[5] http://platypuswiki.sourceforge.net/
[6] http://semediawiki.sourceforge.net/
[7] http://mediawiki.sourceforge.net/
[8] http://rx4rdf.liminalzone.org/Rhizome

HTML or other output formats. Page metadata has to be entered separately from the page. While the approach is very interesting from a technical point of view, the current implementation requires a lot practice with the underlying techniques.

So, current semantic wikis are lacking concerning extraction and usage of metadata—users have to enter metadata manually, and the only means of querying the metadata is either very simple queries built with a user interface or very complex queries entered manually as text in a query language.

3.1.4. Tagging

Tagging means to attach words to resources, like adding "search" to the website www.google.com. Tags are a flat approach to categorization, each resource can have one or many tags, and the tags themselves have no meaning besides there name and the connected resources. The obvious problems are different spellings or different words used for similar items, for example using "Rome" and "Roma" both to identify resources related to the city in Italy.

We can describe a model of tagging as:

- A *resource* is subject of one or many *annotations*. The resource is usually represented as URI.
- A *tag* is a keyword, it is represented as a string.
- The *user* is represented as an entity.
- An *annotation* is a set of *resource*, *tag*, *user*, and *date*.
- Multiple annotations build a tagging system.

There have been ideas around how to connect tagging with the Semantic Web [Gal05, Bec06]. On the Semantic Desktop, we see tagging as user-interface metaphor to allow the user to easily categorize documents using an ontology.

Collaborative tagging and emergent semantics based on tags is also subject of research, under the keyword "*folksonomies*". As this thesis circles around the individual user and does not reach into community aspects, folksonomies are not discussed and using folksonomies in combination with the Semantic Desktop is open for future research handled within the NEPOMUK project.

3.2. Philosophy and Cognition

There are varying definitions of philosophy, their schools, and the exact nature of the schools. A central question is "What is the nature of reality?" and "If things exist, what

is their objective nature?". In philosophy, *ontology* is the study of being and existence, the basic subject of metaphysics. Ontology seeks to describe categories of things, entities that belong to categories, and relationships amongst entities and categories. In early ontology, words and their relation to the real world was analysed. Today, all actors and entities in the whole process are analysed in a scientific way. Language as such (linguistics), the human being and his cognitive system (cognitive science), media, how knowledge is transferred (knowledge management), are detailed fields that refine metaphysics and ontology.

In the next sections, questions are discussed that circle around the question how ontological knowledge is represented by the individual. After this, formal ontologies in computer science are discussed, both influenced our design decisions for the Personal Information Model.

3.2.1. Mental Models

Mental Models have been studied by cognitive scientists as part of efforts to understand how humans know, perceive, make decisions, and construct behaviour in a variety of environments. The term *Mental Model* was discussed by Craik in his 1943 book, *The Nature of Explanation* [Cra43]. It said that humans make use of internal models of external reality, which enable them to better understand and react to situations in their environment. In his view people operate on mental representations to simulate real world behaviour and produce predictions. In other words this implies humans are not just physically situated in their environment, but they also have their own internal model of it, which allows them to deal with the reality of the world.

After Craik, several works on Mental Models appeared. Johnson-Laird's [JL83] theory of Mental Models, and a collection of work on Mental Models of natural phenomena and devices by Gentner and Stevens (1983) [GS83] are further presented.

The Johnson-Laird volume proposed Mental Models as a way of describing the process which humans go through to solve deductive reasoning problems. His theory includes the use of a set of diagrams to describe the various combinations of premises and possible conclusions [JL83]. Johnson-Laird proposes three types of mental representations: (1) Propositional representations, which are pieces of information resembling natural language. (2) Mental Models, which are structural analogies of the world. (3) Mental imagery, which are perceptual correlates of models from a particular point of view.

Another book appeared in the same year by Gentner and Stevens. They proposed that Mental Models provide humans with information on how physical systems work. This approach could be generalized to a number of situations that humans face, including the behaviour of objects according to laws of physics [GS83].

The fundamental philosophical issue addressed within the context of Mental Models is that *the real world (objective) differs from the representation within human thought (subjective)*. For instance, thoughts about an office building are not the building itself, rather a conceptualization based on the perceived features of the physical building, or perceived by reading a description of such features. Additionally, thoughts are not restricted to the subject of existing things, but include things that cannot possibly exist in the physical world (rectangular basketball), things that do no exist (unicorn), things which are not perceivable (limit of universe).

As [CoHF87] pointed out in their introduction, confusion has surrounded the term mental model. We understand the term in this meaning:

A mental model is the mental representation of the real world, that a person creates inside his or her cognitive system. The mental model is created based on perceived input. It can consist of (but is not restricted to) images, language, and relationships. A mental representation cannot be fully externalized by the person, as another person cannot fully reproduce it, nor can an information technology system fully capture a mental model.

Whether the semantic meaning of a mental model reflects facts of the real world, will not be answered by us at this moment, as this is a metaphysical question. Important is that mental models exist and that they can be partially externalized, the aim of our work is to support this externalization process. Constructivism and Gestalt Theory give indications how mental models may relate to the real world.

3.2.2. Constructivism

Constructivism is a philosophic school that is concerned with the question how an individual person perceives, interprets, and then constructs the world, and the relation of the constructed world with the "real world". There are different branches in constructivism, each with a slightly different view.

Formalization of the theory of constructivism is generally attributed to Jean Piaget, who articulated mechanisms by which knowledge is internalized by learners. Constructivism, when applied in learning theory, can also be verified in learning experiments, which (in the eyes of Piaget) makes it attractive for research. He suggested that through processes of accommodation and assimilation, individuals construct new knowledge from their experiences. When individuals assimilate, they incorporate the new experience into an already existing framework without changing that framework. Social constructivism and cultural constructivism interpret the theory further and question the objectivity and truth of certain cultural and social behaviour. For example, some believe that representations of physical and biological reality, such as race, sexuality, and gender are constructed and learned.

Radical constructivism does not deny an objective reality, but states that there is no way to know which constructed reality the objective one is. It can be summarised

as: *Coming to know is a process of dynamic adaptation towards viable interpretations of experience. The knower does not necessarily construct knowledge of a "real" world* [Gla90]. Mental constructs, constructed from past experience, help to impose order on one's flow of continuing experience. However, when they fail to work and do not match the perceived reality any more, the constructs change and try to accommodate the new experiences.

From a radical constructivist perspective, communication need not involve identically shared meanings between participants. It is sufficient for their meanings to be compatible according to [HT97]. If neither of the parties does anything completely unexpected to the other, then their illusions of identically shared meaning are maintained accordingly [Gla90]. The emphasis here is still clearly on the individual learner as a constructor. Neither trivial nor radical constructivism look closely at the extent to which the human environment affects learning: it is regarded as part of the total environment. These issues are focussed on in more detail by social, cultural and critical constructivism. More pointers to literature on constructivism are given in [Dou98].

For the Semantic Desktop, constructivism gives arguments that question and criticise formal, shared ontologies. A real world exists, but descriptions of it are constructed. The idea of a personal information model, which is described later in this thesis, aims to accept this and give the individual a way to express the individually constructed reality in a formal way.

3.2.3. The Semiotic Triangle

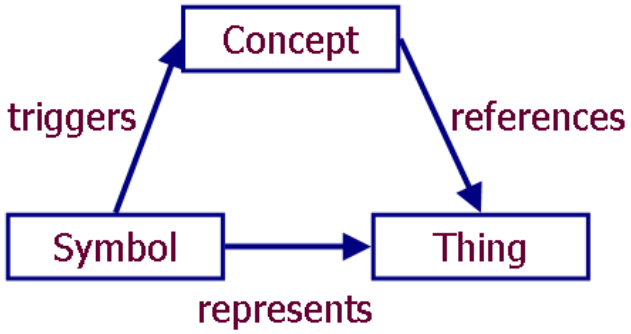


Figure 3.1.: Semiotic Triangle

Another view on mental models, the real world, and representations is the semiotic triangle[9], shown in Figure 3.1. The semiotic triangle illustrates the interaction between symbols (which are typically words), mental concepts, and things existing in the real

[9]It is part of handbooks on linguistic theory, an often cited publication is [OR23].

world. Words which are used to communicate information cannot contain the essential things of the real world. Words and symbols trigger mental concepts in the mind, concepts which are linked (through precedent learning by the interpreting individual) to things in the real world. Ideally, the connections between the corners of the triangle are defined unambiguously. A symbol represents one thing in the world, which is connected to a concept in the user's mental model. Using language, ambiguity exists in synonyms and homonyms. In communication between multiple individuals, misunderstandings can happen.

In communication, the interpretation of a message is up to the receiver. When interpreting, different receivers may connect the message to different mental concepts, based on their training background; in the case of software based on the domain data at hand. That leads to possibly different things of the real world that are associated to a word.

Assuming we have an example user Paul who often has to think about the concept "City of Rome, Italy", then there could be multiple symbols on his computer representing the city, for example a folder called "Rome", a bookmark pointing to the website of the city of Rome, and entering the term "Rome" into google. All three symbolic representations (and actions to find representations) can be connected to the mental concept "Rome", but especially in current operating systems, there are multiple symbols representing the thing. For PIM on the Semantic Desktop, the semiotic triangle is important, as at the moment, multiple or no symbol exists on the personal computer to represent concepts that are in the attention of the user. In this work, we suggest a solution to this problem in the "Personal Information Model".

Ontologies aim now to minimize the possible concepts that can be associated to a word, by describing the concepts using a formal language. Also homonyms and synonyms can be modelled. They enable better communication amongst humans and software agents.

3.3. Personal Mental Models

The philosophic ideas of mental models, constructivism and the semiotic triangle lead us to *Personal Mental Models*, a concept we have published before in [SBD05]. The following text is partly taken from this publication.

Because we do not perceive our environment as a continuum without any intrinsic boundaries, we categorize documents as belonging to (named) classes with (certain inherent) properties. We can verify this by an experiment where a number of persons should categorize a new computer science book or journal article into, e.g., the ACM Computing Classification System (CCS) [Ass98].

Individual Background

Now, let us transfer this idea to the Semantic Desktop where we generate, receive and organize documents. Because of the nature of our brain to classify and store (and

perhaps due to a *hunter-gatherer* mentality), we populate our workspace (and websites, corporate fileshares, etc.) with documents needed to satisfy the daily requirements of our work. This leads to the hypothesis that all documents that are available on our individual workstation are related to our *individual background*, to the ongoing tasks and running processes we are involved in and to our personal interests. Further, the documents capture information about concepts we make of the world: persons, places, projects, topics, etc. These concepts are probably highly subjective but can be expressed using basic application features such as the filesystem's folder structure or enhanced formalizations like OWL ontologies or taxonomies. Documents can be classified using these structures, manually by the user who decides how to classify a document at hand by reading it, understanding it and correlating it to a mental model, or automated by using text classifying engines such as "brainfiler" [MHBR05][10] or GATE [CMBT02]. Hence there exists an interaction between mental models and formal ontologies, mental models find their match in the formal, symbolic representation of ontologies.

Common Background

Although the directories at individual workspaces are highly subjective, we take into consideration that collaborators usually have a *common background*. In [Cla96] it is shown how a shared background and an awareness of a co-worker's activities and mental states contribute to establishing and maintaining communication. This common background has to be expressed using a formalization that addresses the similarities among participating collaborators. If the participants work in a similar topic, then the common background of ie. "biology" may be available in a public ontology, expressed by domain experts, preferably formalized in OWL. Using them allows a sender to describe a message in a category that the receiver will understand, because the same category exists on both computers.

Hence the individual background is expressed using personal mental models, expressed as *personal concepts*; and the common background is represented by *common ontologies*. Both can be formalized as ontologies.

3.4. Ontologies in Computer Science

Gruber has given an often cited definition of ontologies in computer science in [Gru93]: *An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what "exists" is exactly that which can be represented.*

Formalize Information

In information science, ontologies [MSS01] are a way to express knowledge about a certain application domain using a formal model. Ontologies facilitate knowledge sharing, in situations where people and software agents communicate. The term was coined

[10] `http://www.brainbot.de`

in philosophy and is now used in information science, Artificial Intelligence, Semantic Web, and related areas such as knowledge management as well. When working with ontologies, these fields overlap. On the social-cultural side, a group of users have to agree on a formal model of their field, capturing the entities, their properties of these entities and relations. On the implementation side, software agents can read information that is expressed according to these models and perform various operations on this information. In the middle between all of them is the ontology, and data that conforms to it. The main benefit of using ontologies in applications, is that they can be modelled by the domain experts, knowledge engineers, by users of the system, and not necessarily programmers. The results of this knowledge engineering process can be fed into the software to be interpreted. Like UML, XML-Schemas or other data description approaches, the gap between systems and people is getting smaller.

Domain Ontologies

On the *Semantic Desktop*, ontologies are serving multiple roles. The first, in strict interpretation of Gruber's definition, is expressing *knowledge about a domain of interest* and sharing this knowledge *amongst users*. The domain can be a schema of common concepts, for example the FOAF schema is a domain ontology to describe facts about persons and their relations. Wikipedia is evidently a representation of everyday concepts, although not fulfilling all aspects of an ontology (such as rules or a formal representation in every detail), it was used by us to identify and relate common concepts. Abecker and Elst [AvE04] have listed application areas for what I call domain ontologies as knowledge portals for communities of practice, organizational memories, lessons learned archives, skill management systems.

Information Ontologies

Second, and separated from domain ontologies by the fact that they work the level of data, we used ontologies to transform existing data into a coherent representation. For this we created or adapted ontologies to express the information contained in documents, contacts, e-mails, appointments, and multimedia files. The ontologies served *to create a shared data format between applications*, for example the Dublin Core schema is an ontology that describes how data about documents should be expressed. These ontologies are mostly the same on any given desktop, but need adaption for specific applications. In [AvE04] these are called Information ontologies and model what types of documents occur, what metadata attributes they have and what relations amongst documents are represented.

Personal Information Model

Third, ontologies were used to allow a single user to express her or his understanding of the world. There, the sharing aspect was between the user and his or her applications, *subjective knowledge about the world was shared amongst one user and applications*. Summing up, the three areas are:

- **Domain ontologies** are used to conceptualize knowledge about a domain to share it amongst users and systems.
- **Information ontologies** are a standardization of data formats to share data

amongst applications, independent of user or application domain.

- The **Personal Information Model** of a user is the formalization of the subjective view of a user, shared amongst the users and their applications.

These three uses of ontologies can be expressed and used separately from each other, our concern was now to spawn a network of relations between them, forming a unified view and creating connections between them.

3.4.1. Semantic Web and the Resource Description Framework RDF

The idea of ontologies as described above existed throughout the history of information science. In the 1970's, during the rise of expert systems, the golden age of artificial intelligence, formal computer readable languages were developed to express facts about the world. Prolog, frame logic, description logic, they all exist now for quite some years. Now the Semantic Web is a way to sneak in these arcane ideas into everyday knowledge, into the Web.

> "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming."

Semantic Web Architecture

This is the statement (as of January 2007) given at the official website about the Semantic Web[11]. The Semantic Web is an extension of the current web, the basic technologies of URIs for identification, HTML for representation and HTTP for communication continue to exist. Additional to them, more standards are added. They form the "Semantic Web layer cake" architecture, as shown in Figure 3.2. The Resource Description Framework (RDF) is added to describe the documents and information resources available on the web (or in databases) [BLHL01]. On top of that, ontology languages provide meta-languages. SPARQL is the query language to retrieve information. To the right, digital signatures and encryption give the foundation of trust, which is on top. In the next sections, we describe the parts that are important for this thesis.

RDF

The Resource Description Framework (RDF) [FM04] is used as a standard to express information about resources. Resources are first and foremost web resources, HTML documents and multimedia files on the web. Second, they can be things from the real

[11] http://www.w3.org/2001/sw

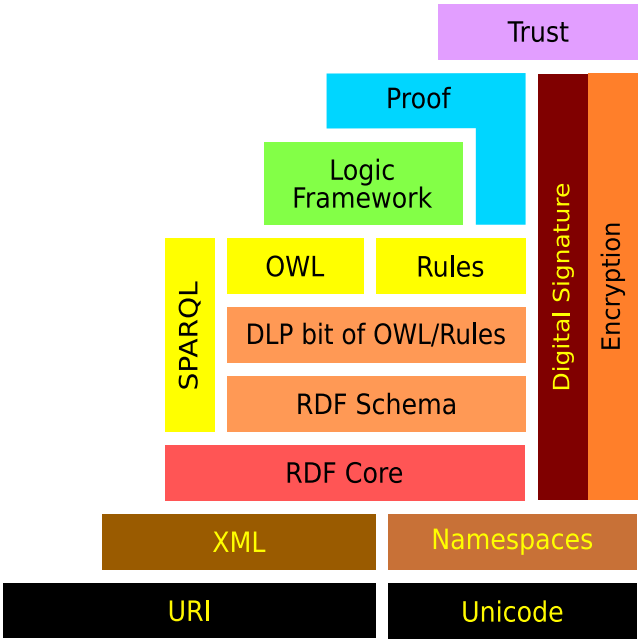


Figure 3.2.: Semantic Web Layer Cake

world, such as products or persons. Facts about these resources can now be expressed using "Statements".

Statements

An *RDF statement* consist of three parts. First the *subject*, a resource that is to be further described. The second part is the *predicate*, what property of the subject the statement further describes. The third part is the *object*, a value of this property. The object could either be a literal (a string value containing a number or a name) or another resource, identified by an URI. Resources can either be URIs or *blank nodes*, which is a resource that has no explicit identifier but is identified by its properties. Predicates are always URIs. The three together, subject, predicate and object, form a statement that is also called a triple. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. To describe the information "This thesis is written by Leo Sauermann", the sentence is divided into subject, predicate and object. The subject is an URI identifying this thesis, the predicate is the authorship and the object is an URI identifying Leo Sauermann. Here is an example:

```
Subject:   http://www.dfki.uni-kl.de/~sauermann/2007/dissertation
Predicate: http://purl.org/dc/elements/1.1/creator
Object:    http://www.leobard.net/rdf/foaf.xml#me
```

These three URIs are globally unique and valid identifiers for the respective concepts behind, the predicate was taken from the Dublin Core ontology. RDF statements can

be serialized as XML files to be exchanged between systems and used for automated processing. There are other notations to write RDF, for example the plain text format N3 [BL98].

Storing RDF Data

RDF data can not only be stored in files, but is also feasible for storage in databases, which are typically called "RDF stores" or repositories. When multiple RDF documents are gathered in one store, there is a need to distinguish amongst the facts (triples) that come from different sources (RDF documents). This is typically done using *named graphs*. A graph is a set of RDF triples, typically coming from one RDF document. Each triple is extended with a fourth value, the *context* or named-graph id. The triples then have four values, also called *a quad*. Stores that support storing these are then called *quad stores* or context-aware stores. With quads and named graphs, there are different ways how the fourth column (the context of a triple) can be used. An overview on different interpretations of quads is given in [CBHS05].

On the Semantic Desktop, facts about domain ontologies, data from the desktop and the personal information model are all represented using the RDF. The data is stored in a RDF store. To support the heterogeneity of data sources and ontologies, a quad-store will be used.

3.4.2. Ontology Languages: OWL and RDFS

RDFS

The W3C provides a recommendation to define the vocabulary which can be used within RDF documents. This vocabulary language is called *RDF Schema (RDFS)* [BG04]. Using it, the structure of RDF documents can be described. Language constructs include classes, subclasses and typed properties. Similar to XML Schema which imposes specific constraints on the structure of an XML document, RDF Schema provides information about the interpretation of the RDF statements. Based on RDF Schema, simple ontology systems can be described. Besides RDF Schema, the Semantic Web standard allows other ontology languages to be created and used.

OWL

The *Web Ontology Language (OWL)* is an ontology language extending RDFS. OWL can further express restriction on membership in classes and restrictions on domains and ranges of properties [MH04]. The formal logic behind OWL is description logic, but there is an innate conflict in this. RDF and RDFS have no restrictions in their logic model, classes can be instances and properties can describe themselves. For OWL, the conflict was resolved by defining subsets of OWL that are logically computable, OWL-DL, OWL-Lite, and OWL-Full. OWL-DL has description logics (DL).

For the Semantic Desktop, which is a user-centered approach, there is no clear and definitive decision which logical foundation should be used. The web ontology group is still discussing (in 2007) a new version of OWL, not necessarily based on RDF, and it seems that we will wait many more years until a logic language and syntax is created that fulfils all the requirements of the Semantic Web community. Until then, we made

the decision to base our approach on the simpler RDFS standard. With a few extensions, such as inverse properties and cardinality restrictions, user-friendly applications can be created.

3.4.3. Topic Maps

XML Topic Maps

Besides logic languages such as OWL or frame-logic, which have their foundation in computer science and logic, an approach to knowledge representation has emerged from the bibliography and knowledge management community. The idea of concept maps and topic maps is a continuation of bibliographical concepts such as thesauri or taxonomies (hierarchical structures). In 1999, this led to a standardization of Topic Maps as ISO 13250 standard and to widespread use under the name "XML Topic Maps" (XTM).

Topics

Topic maps are used to model a domain of interest, one domain is represented in one Topic Map. Subjects in this domain are divided into addressable subjects, which can be stored in a computer, and non-addressable subjects, which are concepts from the real world or imaginary concepts. Both are represented as *topics* in the *topic map*. Topics have names, it is intended that alternate names and lexical variations are stored in the map to find topics easily. Relations between topics are expressed using *associations*. An association can model n-ary relations between many topics, for example all attendees of a meeting can be modelled in one association. Documents from the web or other document management systems can be classified using the topics, the relation is that a topic *occurs* in a document. There are a few predefined association types for common knowledge constructs: subclasses, sub-properties, instances of classes, sub-properties.

Comparison

There are several differences between Topic Maps and RDF. No formal logic such as description logic was used as the foundation of Topic Maps. Instead the needs of document management systems are the basis. Topic Maps carry a found philosophical model behind, that topics are representation of things in the real world, and that documents that describe these things are occurrences of the topic. These map to the things and symbols modelled in the semiotic triangle (see Section 3.2.3), with the interpretation based on the mental model of a user. Topics can be associated with resource, that describe the topic and can serve as identifier of the topic. As an example, the company DFKI is a non-addressable organization. The topic "DFKI" in a topic map can be identified using a website describing the organization, such as "http://www.dfki.de". The identity of the subject is defined indirectly, using the address of the website as indicator, the exact relation is called "subjectIndicatorRef". This is in contrast to addressable topics, "the HTML document retrieved from the address www.dfki.de", the identity of this document is linked to the website using a "resourceRef". The creation date of the document is only a few weeks old, the creation of the organization is the year 1988, both are topics in the topic map, both are linked to the resource "http://www.dfki.de", but using different relations. Topic Maps' precise separation of documents from concepts was included

in the PIMO framework.

3.5. Enterprise Application Integration EAI and Middleware

Middleware

Middleware is, for short, a technology that brings together systems that were never meant to work together. In the *Enterprise Application Integration* (EAI) field, the term "Middleware" describes systems that handle Data Integration, with the goal to access data from a heterogeneous set of systems and make them accessible in a uniform way. Data Integration systems often have an architecture like a wheel, with a central hub and spokes, that connect the hub with the different systems.

Semantic Web Middleware

The Semantic Web provides a data format that can be used for data integration, RDF, and an architectural approach that guides its realization, web services and web architecture. The adaption of relational databases was shown by Bizer, Cyganiak, and Seaborne [CB04, BC06]. In their work, they address the problem of accessing relational databases as Semantic Web endpoints. First, an ontology is needed to represent the existing data. Then, the ontology is mapped to the database. A web-service then exposes the database as Semantic Web endpoint, and the existing data can be accessed through the ontology and the mapping. Queries expressed using SPARQL are transformed to SQL, executed, and their results are again transformed to RDF, all based on the mapping definitions.

Virtual RDF Graphs

Our own contribution the the field of data integration is taking an approach similar to Bizer's "Virtual RDF Graphs" and using it to adapt structured information sources only accessible through APIs, such as the desktop interface to Microsoft Outlook [SS05a]. We have created a framework of adapters that connect to information sources, mapped to RDF schemas using a mapping language. This allows us to extract information from a source on query time, when the content of an e-mail or a file is needed, it is extracted on demand. No copying of data into a buffer is needed, but we noticed a high effort in programming and a slower query answering time. Therefore, for desktop data integration, we concluded to use crawling approaches, copying all data into RDF data stores and do data integration tasks and information retrieval based on this index.

Microsoft Information Bridge Framework

A product by the Microsoft corporation called *Information Bridge Framework*[12] aims in the same direction for conventional data sources: they can be included into office documents via so called *SmartTags*. The framework implements a client–server based approach, the server provides a metadata service that integrated several enterprise web services and other data sources (like CRM systems). The client can be normal office

[12] `http://msdn.microsoft.com/office/understanding/ibframework/default.aspx`

applications, that are extended by plugins: a client gathers current context and keywords from open documents and loads related information from the server.

SECO Integrating web-services is focus of the *SECO: mediation services for semantic Web data* project aiming at integrating web sources [Har04]. It describes an infrastructure that lets agents uniformly access data that is potentially scattered across the Web.

3.6. Summary

In the last section, EAI, we have seen that data integration on enterprise level is feasible and well researched. Taking into account the idea of constructivist philosophy and mental models, we see the need to represent the personal view of users on their data. The Semantic Web technologies RDF and RDFS can provide a standard for this. In this thesis, we will address the topic of personal views using the label *Personal Information Model* in Section 5.

CHAPTER 4
Semantic Desktop Paradigm

How will the current Semantic Web projects lead to the planned global network? Today, many projects in the scientific community around the Semantic Web are focused on enterprise, server or collaborative systems, automated learning of ontologies, synchronization or logic problems. The goals are set on a high level. On-To-Knowledge[1] is a good example for such projects. Its application focus was knowledge management in large and distributed organisations.

The opposite direction was chosen by us: **If the goal is to have a global Semantic Web, one building block is a Semantic Desktop, a Personal Semantic Web for a single user.**

Personal Computers

Nearly all information we see on web-pages and in electronic documents today has been created by people using personal computers. The PC is the place where most personal data is stored and the major interface to the web. Information stored on a server is usually manipulated through interfaces that are executed on a PC. To create all this content, we need to bring the Semantic Web to the personal computer, to the person in front of the screen. The use of Semantic Web technologies on a desktop is a step towards the Semantic Web. Ontologies, classifications and global identifiers can create an immediate benefit for personal information management. If users can benefit from creating Semantic Web content on their desktop, because they find information more quickly or can organize their work better, the data created this way is a viable input for the Semantic Web. On the other hand, data taken from the public or corporate Semantic Web can be integrated on the Desktop without a glitch in technology, seamless, if the Desktop is Semantic-Web enabled.

gnowsis

The first research project targeting a Semantic Desktop system is the author's *gnowsis Semantic Desktop* [Sau03] diploma thesis. The goal of gnowsis is to complement established desktop applications and the desktop operating system with Semantic Web features, rather than replacing them. The primary use for such a system is Personal Information Management (PIM), technically realized by representing all data in RDF. The thesis addresses the problems of how to identify and represent desktop resources in an unified RDF graph. The project was released under an open-source license and was the basis for other research projects.

EPOS

In *EPOS* [DAB+02], a service oriented architecture based on platform-independent

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

communication protocols (XML-RPC) was added to gnowsis. Also, the idea of a *Personal Information Model* [SDvE+06] was created and used to express the personal view of the user.

4.1. Related Work

To allow users better annotation of documents from within existing applications, such as MS-Word, Adobe Acrobat, or the web-browser, Iturrioz, Anzuola and Díaz have turned the mouse into a semantic device in their *seMouse* project [IAD06]. The idea is that independent of application, the user can annotate the document by pressing the middle mouse button, or text elements of the document by selecting them and then pressing the middle mouse button. The simplicity of this approach is striking and their prototype implementation was tested in combination with the gnowsis prototype.

The *@Visor* [EAD+06] project realised visualisation and navigation techniques in semantic virtual environments, a three-dimensional interface was evaluated. The architecture and approach from these projects was described in [SGK+06].

A major research project concerning an integrated approach in our field is the *Haystack* system by Quan et al. [QHK03] from the MIT Computer Science and Artificial Intelligence Laboratory. It is an integrated approach to let an individual manage her information in a way that makes the most sense to her. Application-created barriers of information representation and accessibility are removed by simply replacing these applications with Haystack's word-processors, email client, image manipulation, instant messaging and other functionality. They provide a complete semantic programming environment, from user interface to database.

Also based on RDF technologies, but with a pragmatic approach of keeping the complexity to the necessary minimum, is the *MetaDesk* project [MMY04]. The user can freely annotate files and folders, instead of using formal ontologies and complex autogenerated user interfaces the authors chose partonomies (part-of) and set membership as semantic structures for categorization. Plugins are used to edit particular data types (instead of completely dynamic applications as shown in Haystack). The authors have shown that their pragmatic approach can come to results comparable to Haystack, but with much less effort.

Another relevant personal information management tool is the *Semex System* (SEMantic EXplorer) [DH05], which organizes the data in a semantically meaningful way. Users interact with Semex through a domain ontology that offers a set of meaningful domain objects and relationships between these objects. Semex employs multiple modules for extracting associations, as well as allowing associations to be given by external sources or to be defined as views over other sets of associations. Semex aids user information integration tasks by trying to leverage from previous tasks performed by the user

or by others with similar goals. Hence, the effort expended by one user later benefits others

The idea of the personal information management system *IRIS* [CPG05] is to have an integrated environment, similar to Haystack, but based on standard software. An e-mail client, calendar and other productivity tools are integrated into one coherent interface, allowing to classify and display related information. Algorithms provided by the CALO cognitive assistant analyse the content of documents and find related documents and possible classifications. The assistant learns over time. This project shows how artificial intelligence can be embedded in personal information management.

The community aspect of Semantic Desktop applications was addressed in the DBIN project [TMN06]. DBIN is a distributed system, based on a peer-to-peer Network, that allows editing Semantic Web content within groups. In this way each client builds and populates a 'personal semantic space' on which user defined rules, trust metrics and filtering can be freely applied.

Apart from the purely scientific projects, we see individuals and organizations working on implementations of systems that fulfil many aspects of the Semantic Desktop.

The *Chandler system* is an interpersonal information manager, adapting to the changing user needs. It was initiated by Mitch Kapor, who founded the Lotus Development corporation, which created the very popular and innovative Lotus Notes system. Chandler delivers an integrated system for individuals and small work groups, offering knowledge sharing capabilities to support work group collaboration. It allows users to share collections by publishing them to a server and similarly, a Chandler user can subscribe to other users' collections. The domain model defines all of the domain specific classes that represent application content such as Calendar Events, Mail, Messages, Tasks, etc. It simplifies information sharing with others, and calls itself an *Interpersonal Information Manager*.

Ontooffice by ontoprise—a corporation close to semantic web research—is a desktop product that brings together the contents of a semantic web server and Microsoft Office applications. The scenarios are similar to those of Microsoft's SmartTags and the Information Bridge Framework.

Martin Dvorak has written the *Mindraider*[2] open source software. It is an application for personal note-taking, using RDF as data structure. The outstanding point of Mindraider is that it combines classifying information in a tree (as we find in file-systems) with a graph-navigation view. Each document or information item can be viewed as part of a networked graph, the software shows the relations and connected elements. The project has not seen much development in 2006.

The *Fenfire* [3] project is at an earlier stage, dealing with the problem of visualising and

[2] `http://mindraider.sf.net`

[3] Last Fenfire release in Jan 2007.`http://fenfire.org/`

editing RDF graphs in a uniform way. It is a completely based on RDF and implements various user interface metaphors. Parts of the system are published, others are kept closed because of patent issues.

Another approach was taken by Joe Geldart in his bachelor thesis about the *frege* system [Gel05]. He describes a minimal implementation of an RDF desktop communication framework on which a few example applications are implemented. The thesis tackles the core ideas and finds a minimal and efficient solution. The idea to use DBUS as communication system later appears again in Nepomuk's KDE project.

Integrating *ontologies* for personal systems, and proposing a layered approach to ontologies was done by Huiyong Xiao and Isabel F. Cruz [XC05]. They differentiate between *Application Layer, Domain Layer and Resource Layer*. We will find a similar architecture of layered ontologies in NEPOMUK.

In addition to Semantic Desktop projects, we can integrate various other interesting ideas such as *Semantic Wikis* [Sau03, KS05] or *Semantic Blogging* [Cay04], or Semantic E-Mail [MEHL04] to improve PIM.

4.2. Analysis

What we see in these projects are well-founded research on different aspects of the Semantic Desktop. In Haystack we see all the elements needed for an adaptive, user-friendly user interface. On the other hand, it is slow and needs vast system resources. This can be improved by rewriting the project, but only an industry investment would give reason to do this, for science the processing time a GUI needs is not interesting. In DBIN we find a solution to split RDF graphs into small parts to be able to identify and distribute them via a peer-to-peer network. But the user interface of DBIN lacks a proper evaluation. In OpenIRIS we see how a cognitive assistant works, but no peer-to-peer interaction takes place. We could go on now, pointing out that each project can only address one particular problem at a time. The overall problem is, when doing an evaluation with test users, the users will immediately see the problems. When we want to evaluate only the p2p aspect, but lack a good user interface, the test users will mock about the user interface, and an evaluation of the interesting part is influenced.

4.3. NEPOMUK

Nepomuk

The EU Integrated Project *NEPOMUK* aims at connecting several open source projects, and do fundamental research in all areas involved: data integration, peer-to-peer, social collaboration, user interfaces and more. The scientific results of the project are evaluated within industry settings in four case studies. The approach is implemented using

open source prototypes, so that other researchers can verify the results and build upon them. This project was initiated by Stefan Decker and myself to create a focal point for the Semantic Desktop. Lead by the DFKI, this EU IST integrated project (IP) brings together research partners from NUI Galway, EPFL Lausanne, DFKI Kaiserslautern, FZI Karlsruhe, L3S Hannover and ICCS-NTUA Athens with practitioners from companies like HP, IBM, SAP, Mandriva, Thales, PRC Group and others, building a community of experts. NEPOMUK bundles academic, industrial and open source community efforts to create a new technical and methodological platform: the *Social Semantic Desktop*. A number of case studies apply, adapt, and test NEPOMUK's solutions in various knowledge-work scenarios. NEPOMUK's standardized plug-in architecture combined with usage experiences opens up manifold business opportunities for new generic or domain-specific products and services. Using the methodology that spread the World Wide Web – *open standards, open source reference implementations and continuing communication with the global developer community* (as described in [Lee00]) – the Semantic Desktop community at large gains momentum through this project.

4.4. Definitions

The path was sketched when Stefan Decker and Martin Frank stated the need for a "Networked Semantic Desktop" [DF04] in 2004. Decker recognized that several new technologies had emerged which could dramatically impact how people interact and collaborate: The Semantic Web, P2P computing, and online social networking. He presented a vision of how these different thrusts will evolve and produce the Networked Semantic Desktop, which *"enables people and communities to directly collaborate with their peers while dramatically reducing the amount of time they spend filtering and filing information"*. A roadmap leading to it was created as follows:

- In a first phase, Semantic Web, P2P, and social networking technologies are developed and deployed widely.

- In the second phase, a convergence between the existing technologies brings Semantic Web technology on the desktop leading to the Semantic Desktop. In parallel, Semantic Web and P2P are incorporated and lead to Semantic P2P. Social networking and Semantic Web lead to ontology driven social networking.

- In a third phase, the social, desktop and P2P technology fully merge to a *Social Semantic Desktop*.

Based on the previous publications [Sau03, DF04, Sau05] and using a similar wording as Bush [Bus45] we have first defined a Semantic Desktop in [SBD05][4]:

> A Semantic Desktop is a device in which an individual stores all her digital information such as documents, multimedia and messages. These are interpreted as Semantic Web resources, each is identified by an URI and all data is accessible and queryable as RDF graph. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems, and Semantic Web protocols are used for inter-application communication. The use of Semantic Web standards allows existing web resources to be incorporated into the personal knowledge space, and does also facilitate the sharing of knowledge with others, for example within a work-group. The Semantic Desktop is an enlarged supplement to the user's memory.

The Semantic Desktop can be the driving paradigm for desktop computing in the area of the Semantic Web. It bridges the gap between corporate data management and the knowledge workspace of individuals.

In this chapter the existing work of others was presented. In the next chapter the core elements of a Semantic Desktop, in our understanding, are presented and discussed thoroughly.

[4] Here is a slightly rephrased definition, a spelling mistake was corrected and the wording changed, if you want, refer to this definition for future publications

II

Building the personal Semantic Web for PIM

CHAPTER 5
The Personal Information Model

Weaving the Personal Semantic Web

In this part of the thesis the necessary architecture of a Semantic Desktop is presented, together with details about the *gnowsis* prototype implementation. For the user, the main benefit of having a Semantic Desktop is to be able to create a **personal information model (PIMO)**. It is a data model which captures the personal view of a person on the surrounding world. We employ the Semantic Desktop principles outlined in the introduction. **Files** on the user's filesystem are identified using URLs and annotated with RDF. The personal knowledge space 2.1 goes beyond files, it encompasses all kinds of **resources**: e-mails, appointments, address book entries, todo lists, and more items that appear interesting for *Personal Information Management*. All these items are identified with URLs and their data is represented using RDF. In Section 5.2 we define the **personal information model** as a personal ontology that is used to represent persons, places, documents and projects, shared across applications. How the PIMO is realized is presented in Sections 5.3 and 5.4. We will show how the PIMO is a comprehensive whole, created from a mixture of existing data, personal annotations, and data extracted from various desktop data sources. Applications can use the PIMO to **tag items**, create free-text annotations using **a personal semantic wiki** (Section 5.5). The **limitations and problems** of the PIMO model are discussed at the end of this chapter, in Section 5.7.

The software components realizing this architecture are provided as services on the desktop, usable from various applications via remote procedure calls. This allowed us to build stand-alone applications supporting PIM or to add plugins to existing applications. A description of these services and the applications is provided in Section 6.2. In concert, the background services, the stored data, the semantic meaning of this data, and applications working on this data form the **personal semantic web**, which we implemented as described in Chapter 7.

In the next part, "Case Studies", we show how the personal semantic web augments Personal Information Management in practical examples.

5.1. The example data: Paul and Rome

> Grau, teurer Freund, ist alle Theorie, Und grün des Lebens goldner Baum.
> *Mephistopheles in Faust I, Goethe, 1808*

A common German proverb from Johann Wolfgang von Goethe's drama "Faust" is: "*grey, dear friend, is all theory, and green life's golden tree*". As humans enjoy colour, a prototype implementation of the Semantic Desktop was built, the gnowsis system. References to its architecture and implementation will be given throughout the next sections. Various experiments were concluded to test if our design can be implemented, how to implement it efficiently, how to improve performance, and how users can benefit from it. The implementation details are described in Chapter 7.

Paul and his Office in Rome

To better illustrate the following sections for you, and to focus ourselves to the needs of real users, we have created the **example user Paul**. Paul works together with Tim and Peter on the task of opening a new branch office in Rome, Italy. Paul exchanges a few e-mails about this topic and organizes a meeting about the project. This example scenario is also represented in RDFS, you can find it in the Appendix A.2. [1]

Paul is a knowledge worker and handles information from various information sources, local data sources like files stored on his desktop computer or contacts in his local address-book, organizational data from shared file repositories or ontologies and web resources.

During the creation process of the following ideas, Paul was used in discussions, implementations, and to test ideas against the scenarios created around this person.

5.2. Introduction on the Personal Information Model

The Semantic Desktop requires a well-thought information model. Existing ontology languages like RDF/S, OWL, SKOS and XML Topic Maps are very well suited for certain application areas, but need adaptations to support a single user doing Personal Information Management. In this chapter a new vocabulary, extending RDFS, is proposed: the **Personal Information Model (PIMO)**. A PIMO [2] is used to represent a users' concepts, such as projects, tasks, contacts, organizations apparent in daily work, which then are used to categorize files, e-mails, and other resources of interest. This concept-based categorization integrates information across different applications and file formats.

[1] For the curious reader: the creation of Paul was inspired by the letters of the Apostle Paulus to the Apostle Peter and their apprentice Timotheus.

[2] The abbreviation PIMO does not mean "Personal Information Management Ontology", as previously referred to in [Sau06].

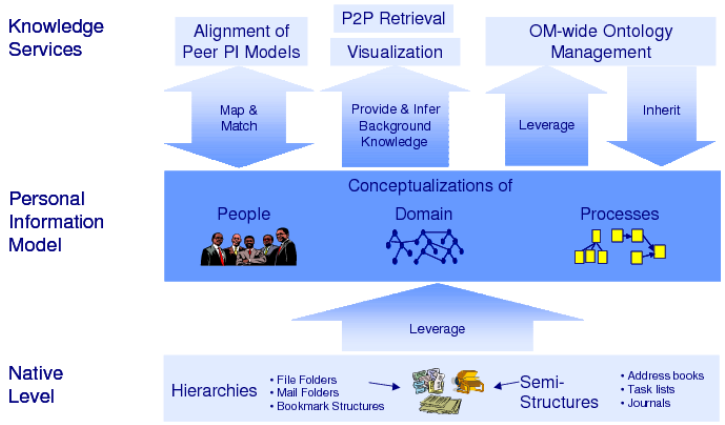


Figure 5.1.: The Personal Information Model as *semantic middleware* between native structures and knowledge services.

Based on RDF/S, multiple layers were defined: an upper-layer containing a minimal set of generic concepts, a mid-layer for refinements, and a user-layer for concepts of the individual user. The PIMO helps users to categorize resources for Personal Information Management (PIM), it is the integrative part in Semantic Desktops, as shown in Figure 5.1.

Supplementary to the description of the ontology is a RDFS version, created using the Protégé tool. An example mental model is given for the fictional user "Paul". The approach was published first in a technical report [Sau06], then used for personalization [SDvE+06]. In 2007, another publication followed [SvED07]. The ontology evolved slightly, for further reference and practical applications we suggest reading the guide to the ontology framework which was published as result of the NEPOMUK project [SEM07]. Parts of this thesis are also published in this guide.

5.2.1. Motivation and Input

Based on studies about file management we know about the importance of native structures (folder hierarchies, etc.) for finding and reminding information [BN95]. A model for PIM should allow using multi-perspective classification [Den06b]. There has been research in synchronizing different native structures of applications with each other in a n:n approach [Boa04]. A unified model removes the need to interact with all other applications and lowers the integration costs by allowing each application to integrate with the model. For the *Semantic Desktop*, where Semantic Web technologies are already used, the PIMO is intended as a cornerstone for data integration. Where today each application brings a separate classification scheme (the filesystem folders, tags, color

tags, positions, mail folders), using PIMO the semantics behind these structures can be captured. In the future, we expect that applications do not need to manage classification schemes on their own but rather use PIMO structures.

PIMO is based on the idea that users have a *mental model* to categorize their environment[3]. We used the following existing approaches as input:

- First and foremost, the Enquire information management system by Tim Berners-Lee. This system is, from the semantic ideas and architecture, quite close to PIMO. Some parts-of its ontology are resembled in our PIMO-language. Enquire is the predecessor of the WWW, of the Semantic Web. One of our goals is to bring Enquire back to the Semantic Web. Luckily, the manual of Enquire is on-line for our reference [BLP80].

- All features of RDFS are used [BG04]. From the OWL ontology language we employ inverseFunctionalProperties and inverseProperties [MH04].

- A paper on 10 years of practical experience in PIM [Roh05] in which by Jean Rohmer reports that inverse-properties are mandatory for every relating property (as does Enquire).

- A paper by Huiyong Xiao and Isabel F. Cruz describing a Multi-Ontology Approach for Personal Information Management [XC05]. They suggest to use different ontologies for the resources and concepts and then map among them; they differentiate between *Application Layer, Domain Layer and Resource Layer*. In PIMO, the resource layer is modelling the same as in their work, application and domain layer are both merged in the mid- and upper-layers of PIMO.

- Alexakos et al. described "A Multilayer Ontology Scheme for Integrated Searching in Distributed Hypermedia" in [CAL06]. There, the layers consist of an *upper search ontology layer, domain description ontologies layer, and a semantic metadata layer*.

- The SKOS ontology language and the ongoing SKOS effort. It models concepts and narrower/broader relations between concepts, allowing to represent concept schemes as a tree of concepts. Type-relations, subclass, part-of, and sub-topic relations all map to narrower/broader. SKOS also provides alternative labels for resources [(ed04].

- The XML Topic Map standard (XTM). It offers — examined philosophically — much more ways to express mental models than OWL or SKOS allow. Concepts

[3] Which is based on the assumption that our users are rational thinkers :-)

are represented separated from documents, which are occurrences of the concepts (a document has a concept as topic). Associations between concepts have a similar semantic as in SKOS but can be n-ary. One association can connect multiple concepts, whereas in RDF the relations are binary. Also, associations can be annotated (for example, having a scope or a start and end-date). Mapping multiple ontologies is possible, based on multiple identification techniques. XTM knows direct and indirect identifiers and allows to use "well-known" documents as references to the topics they cover. The usability of topic maps was proven in many products that are sold on a growing market [SP01, Rat03].

- The wikipedia project was used as an inspiration for standardization and reuse of information.

Representation

A possible approach to combine these input sources would have been to reuse parts of the ontologies by directly copying their defined predicates and classes into our model. This "copy useful bits" approach has also driven the popular FOAF vocabulary (which uses RDFS modelling, but parts of OWL to represent the ontology as such and subclass to wordnet) and vocabularies around it. The mixture of vocabularies would imply, on an RDF-aware user agent, that the class person is also a word, which may be misleading.

To represent entities and their relations, different approaches are used. RDFS which can represent classes and subclass relationships, properties and subproperty. OWL models many facts on the class level, and description logic allows testing facts by checking if an entity is member of a class. Contrary, there are standards to describe mind maps, XTM and SKOS. We cancel out XTM now, as it is not RDF. In SKOS, each entity has the same class: concept. Hence, we cannot model properties and domain/range restrictions - which property may be useful on which kind of entity. Using SKOS it is not possible to say that a person has a name. Also, SKOS replicates RDF's "type" relation with "narrowerInstantive".

Another aspect is the separation of concepts from documents. XTM has a clear semantic separation of those, in RDF there is none on the language level, leading to much confusion about the question "what is a good URI for a concept"[4]. Instead of mixing RDFS, OWL, SKOS, and XTM we chose to rather start a new vocabulary and wait for more best practices to evolve.

As part of the result, we have modelled an upper ontology consisting of a handful of general concepts that appear in many user's PIM systems. Such concepts are indepen-

[4] A confusion which appears with stunning perseverance every two months on the public semantic web mailinglist, as a hobby I marked my answers in these discussions with the keyword "uri crisis". The issue was not solved on a semantic level in RDF, but decided by the Technical Architecture Group of the W3C on the HTTP protocol level. We explain the solution in an tutorial [SCV07]. Still, there is no broad agreement how to annotate a URI in RDF as "this URI identifies a concept and not a retrievable document".

dent of application domain, country or culture of the user, language, and software. The upper ontology was modelled based on our own interviews with knowledge workers, but many other authors did come to similar results[5]:

- The *User Profile Ontology* version 1[GKV+06], mentioned in [CDK+07].
- A list of top level concepts for categorization in graphical user interfaces was proposed: geographical location, alphabetically, by time, by categories, by hierarchy. The combination of these classifiers is known under the acronym *LATCH* [WSLW01, p40].
- Latif and Tjoa describe a similar approach in 2006, where they start with LATCH and come to similar top-level categories [LT06].

The sheer amount of input taken into account indicated that we needed to combine multiple ideas into one model. From the related work, the most important input **requirements** have been:

- a representation of abstract concepts: Love, Profit, Acme Incorporated.
- a representation of addressable resources: ”w3c homepage at www.w3.org”
- a representation of documents: ”the document at http://www.w3.org/”
- multiple names for a thing: ”Love”, ”Liebe”; ”W3”, ”WWW”
- same name for two different things: ”Apache - helicopter”, ”Apache - software”.
- class-subclass relations: a subclass can have all properties of the superclass and its own
- class-instance relations
- part-of relations: the city of Rome is part of Italy
- related information: Spaghetti is related to Italy
- inverse relations are mandatory for related information: Italy is related to Spaghetti
- N-ary associations

[5]The idea appeared repeatedly. As a historical footnote, we want to mention that Jerome Euzenat proposed a top-level ontology for PIM in the light of FOAF on a meeting in 2002. `http://www.w3.org/2001/sw/Europe/200210/calendar/SyncLink.html`

- data properties to describe details: Rome has a population of 2.8 mio
- document-has-topic: the document ”http://www.w3.org/2001/sw” is about the ”Semantic Web”
- distinguished identifiers to match things across multiple PIMOs
- means to map a user's PIMO to a public ontology or to other person's PIMO
- a representation of time: the document was created in 2005. The project started on 1.1.2006

It is not the scope of PIMO to create a new ontology scheme representing data about all the elements a user can possibly work with. Rather, it represents the user itself and the fact that he has a personal model. The design rationale is to keep the PIMO ontology *as minimal as possible*, and also the data needed to create a PIMO for a user as minimal as possible. Inside one PIMO of a user, duplication is avoided. Instances should be represented once, the same with classes. To reach this goal, other ontologies are used to represent information elements (files, e-mails, appointments, etc.) on the desktop.

5.2.2. Definition of a PIMO

Based on the definition of terms given in the introduction in Section 2.2, a definition for a Personal Information Model can be given.

Definition: *A* PIMO is a *Personal Information Model* of one person. It is a formal representation of parts of the users *Mental Model*. Each concept in the Mental Model can be represented using a *Thing* or a subclass of this class in RDF. Native Resources found in the *Personal Knowledge Workspace* can be categorized, then they are occurrences of a Thing.

The PIMO is a formal representation of the structures and concepts an individual knowledge worker needs, according to her or his personal mental model. It is an application-independent and domain-independent representation. Concepts used to categorise elements in one application will also appear in other applications.

The vision is that a *Personal Information Model* reflects and captures a user's personal knowledge, *e. g.*, about people and their roles, about organizations, processes, things, and so forth, by *providing the vocabulary* (concepts and their relationships) for required expressing it as well as concrete instances. In other words, the domain of a *PIMO* is meant to be “all things and native resources that are in the attention of the user when doing knowledge work”. Though “native” information models and structures are widely used, there is still much potential for a more effective and efficient exploitation of the underlying knowledge. We think that, compared to the cognitive representations humans build, there are mainly two shortcomings in native structures:

- *Richness of models*: Current state of cognitive psychology assumes that humans build very rich models, encoding not only detailed factual aspects, but also episodic and situational information. Native structures are mostly taxonomy- or keyword-oriented.
- *Coherence of models*: Though nowadays (business) life is very fragmented humans tend to interpret situations as a coherent whole and have representations of concepts that are comprehensive across contexts. Native structures, on the other hand, often reflect the fragmentation of multiple contexts. They tend to be redundant (i.e., the same concepts at multiple places in multiple native structures). Frequently, inconsistencies are the consequence.

The *PIMO* shall mitigate the shortcomings of native structures by providing a *comprehensive model* on a *sound formal basis*. A *PIMO* should be exploitable by formal knowledge services and provide bridges and transitions to organizational knowledge management systems. As the information in a knowledge worker's individual information space (e.g., his file system) can become valuable knowledge in his tasks, integration of this information with his *PIMO* is especially important.

Coverage Limitation of a PIMO A Personal Information Model should only cover the entities that are relevant for one user for individual knowledge work. By limiting the model to concepts needed by one individual, the amount of information to be represented in one model is limited to a lower number. The theoretical aspects of limiting the boundaries of a PIMO are covered in Section 5.7.2.

In the following, we describe how the concept of a *PIMO* has been realized within the EPOS, gnowsis, and NEPOMUK projects.

5.3. Realization of the PIMO Approach

In this Section, and the following Sections, the technical realization of the conceptual Personal Information Model will be described.

From the ontology designer's view, we distinguish between elements that are modelled by other's (the representational language), stable upper classes defined by us and forming a reusable framework for knowledge modelling (the upper classes), shared knowledge models provided by the organization (company) of the knowledge worker, and the individual model of one user, primarily modelled by himself.

In the next Section, this distinction will be further explained. Section 5.4 then shows how the PIMO parts are applied for one individual user. In this application-oriented part, the different PIMO parts are presented in order of use, so the upper classes are not presented up front, but when they are needed.

5.3.1. Parts of the PIMO

When building concrete *PIMO*s, we now have the problem of two, potentially conflicting demands: On the one hand, we want to give the user the opportunity to span his information space largely in the way *he* wants. The *PIMO* should model *his* mental models. Using a multi-layer approach (see also [SGK+06]), we distinguish between different layers in the PIMO framework, as shown in Figure 5.2. On top of the Figure you find PIMO-Basic with representational aspects, PIMO Upper-Level with globally shared classes, PIMO Mid-Level which are shared ontologies, and the PIMO of the individual person (in the picture the person Paul is represented). This separation is in harmony with the authorship: the upper and representational parts are predefined by us, the shared parts by the organization, and the individual parts by the user, see also Figure.

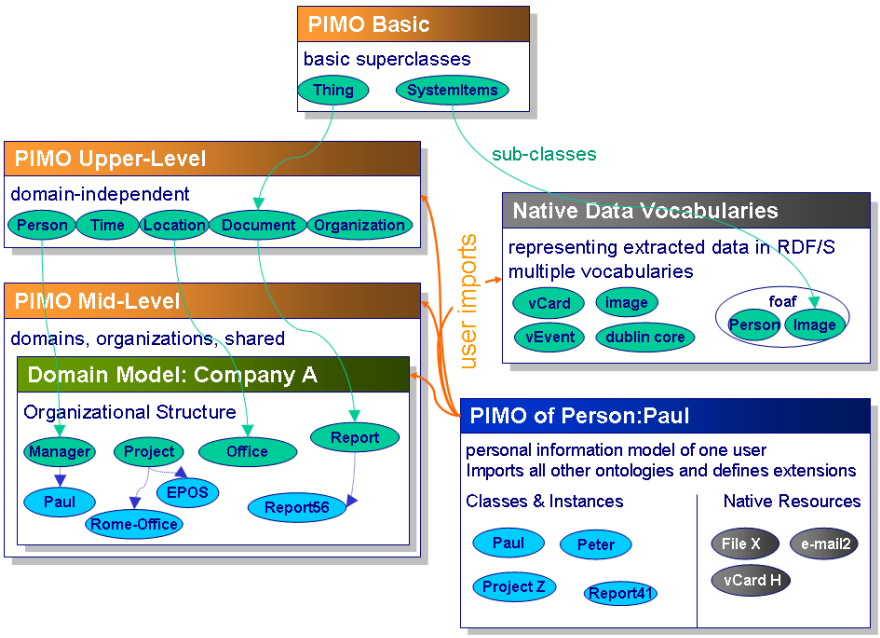


Figure 5.2.: PIMO ontology components

Predefined by the model

- PIMO-Basic: defines the basic language constructs and representational layer. The class pimo-basic:Thing represents a super-class of other classes.
- PIMO-Upper: A domain-independent ontology defining generic sub-classes of Thing. Such classes are Person, Organizational, Location, Document, etc.

In PIMO-Basic, two key concepts are introduced: the classes *Thing* and *Information Element*. *Thing* is a superclass of abstract concepts and physical objects, with the aim of representing them on a conceptual level. *Information Element* is a class to represent the documents in a computer system. To represent these, the *NEPOMUK Information Element Ontology*[6] can be used. The *native structures and resources* can be transformed to RDF as presented in [CB04],[SS05a]. They are represented using subclasses of *Information Element*.

The idea is that a `Thing` can now occur in one or many resources. This is represented by a `occurrence` relation. More relations are described below, in Section 5.4.6.

The *upper ontology* defines common sub–classes of `pimo:Thing`. These classes are independent of culture and application domain, they are listed in Section 5.4.13. Generic relations between Things are also defined, they are listed in Section 5.4.14. More specific relations extend these generic relations, see Section 5.4.15.

Created by the individual

- User-PIMO: the extensions of above models created by an individual for personal use. Classes, properties and things are created by the user.

The mid and upper dimensions of a *PIMO* are pre-given, but flexible enough for simple as well as more advanced modelling. They are meant as a proposition; when establishing an individual *PIMO*, users *can* incorporate them into their model if they find them adequate and useful, but they don't have to.

Shared models provided by organizations

- PIMO-Mid: More concrete sub-classes of upper-classes depending on the user's interests, organization and work topics. The mid-level ontology layer integrates various domain ontologies and provides more specific classes than Person, Project, Company, etc. Mid-level ontologies are shared amongst users.

- A domain ontology describes a concrete domain of interest of the user. The user's company and its organizational structure may be such a domain, or a shared public ontology. Classes are refinements of PIMO-Upper or other mid-level ontologies, allowing an integration of various domain ontologies.

In the next Section, more details about the acquisition of shared ontologies is given. Then, creating a model for an individual is presented in Section 5.4.

[6] `http://www.semanticdesktop.org/ontologies/nie/`

5.3.2. DFKI-KM-Mid: Acquisition of an Exemplary PIMO Mid-Level

The upper level of a *PIMO* just makes a few, basic ontological statements about things which exist on a Semantic Desktop, *i. e.*, things which are essential in a knowledge worker's mental model: *Information elements*, *people-*, *organization-* and *process-related things*, but of course also basic ontological categories like *space* and *time* concepts well-known (and imported) from other typical upper-level ontologies. Obviously, the commitment in this statement is very fundamental for the concept of a Semantic Desktop, but also very abstract. In order to avoid a *cold start problem*[7] with PIMO-based applications, we pre-modelled a PIMO-Mid-Level as a refinement of the upper level which serves two purposes: Firstly, the concepts of the mid level serve as *anchor points* for a user's personal incremental extensions of his PIMO. For example, having already a couple of *project types* as examples in his *PIMO* (instead of just having projects as abstract organizational concepts) makes it probably much easier for him to classify already existing projects or to model new project types. Moreover, offering a common mid level layer to a *group of people* can also be seen as a *seed for a shared conceptualization* between these people, facilitating information exchange on the basis of these shared parts of their *PIMO*s. So, conceptually, the *scope* of a *PIMO* mid-level is a *group of user's* who potentially share many concepts on their Semantic Desktop (*e. g.*, people in the same department), while the control with respect to extensions or modifications is intended to be at the individual user.

For our prototype and the EPOS project, Ludger Van Elst (supported by Laura Zilles) modelled an exemplary *PIMO* mid-level using the following methodology, consisting of the three phases *seeding*, *reality match*, and *evolution*:

In the *seeding phase*, a couple of exemplary *native structures* (file and email folders) of members of DFKI's Knowledge Management Department were manually analysed and so laid the basis for an *initial DFKI-KM-Mid model*. DFKI-KM-mid mainly consisted of concepts without deeper modelling, like attached slots etc.

In the second phase, this initial model was *checked* by a detailed survey. 23 members of the department were interviewed whether the initial model fit their individual native structures, which concepts were missing in the model or not occurring in their native structures

The results from the reality match were used for *evolving and extending* the DFKI-KM-Mid model. Further extensions have been made by a more detailed modelling of slots and by the integration of third-party ontologies like FOAF and specially tailored

[7] The problem of cold starts is very well known in knowledge-based systems: In the beginning a system, like a shell, just has little of no information and therefore seems not to be useful to a new user. Consequently, he is not motivated to invest in using and feeding the system with new information which would be a prerequisite to be *more* useful.

domain ontologies like the "*Organizational Repository*", formalizing the employees and projects of the DFKI KM lab.

Here the idea is that when bringing the *PIMO* idea into a specific environment the mid level should be re-modelled in a similar way as described above. In section 11 we show an example for that in a concrete business scenario.

5.4. Modelling a User's PIMO

Using above prerequisites, the *Personal Information Model* of a user can now be created by assembling the different parts. We will use the example user *Paul* and **Paul's PIMO**, which is included in appendix A.2. In this application-oriented part, all PIMO modelling concepts will be presented in order of use. For example the upper classes are not presented up front, but when they are needed to find a type for a Thing.

The following steps are necessary:

- The user as such and his PIMO are represented.
- The PIMO vocabulary including the basic and upper classes is imported unchanged.
- One or more mid-level ontologies are imported, *e. g.*, the "Organizational Repository" of a company.
- The user and applications can freely extend the model with new things, classes of things, define new properties, or import existing schemas.
- Native resources are integrated and categorized, categorization schemes are learned.

Hence, the *Personal Information Model* (PIMO) of a user can be defined as the sum of imported upper and mid-level ontologies, one personal mental model of the user (user-PIMO), and the native resources found in heterogeneous data sources categorized by the conceptual structures.

As an example for a project managed by Paul, we assume he is planning to open a branch office of his company in Rome, Italy. This project is represented as `paul:BranchOfficeRome`, an instance of class `pimo:Project`. To express that the co-worker Peter is part of the project, `paul:Peter` was created and related to the project via the `pimo:hasPart` relation. Peter has a grounding occurrence in the address book of Paul, the address book entry is an information element. The example goes on to create a custom class (`paul:BusinessPlan`) and custom properties (`paul:manager`).

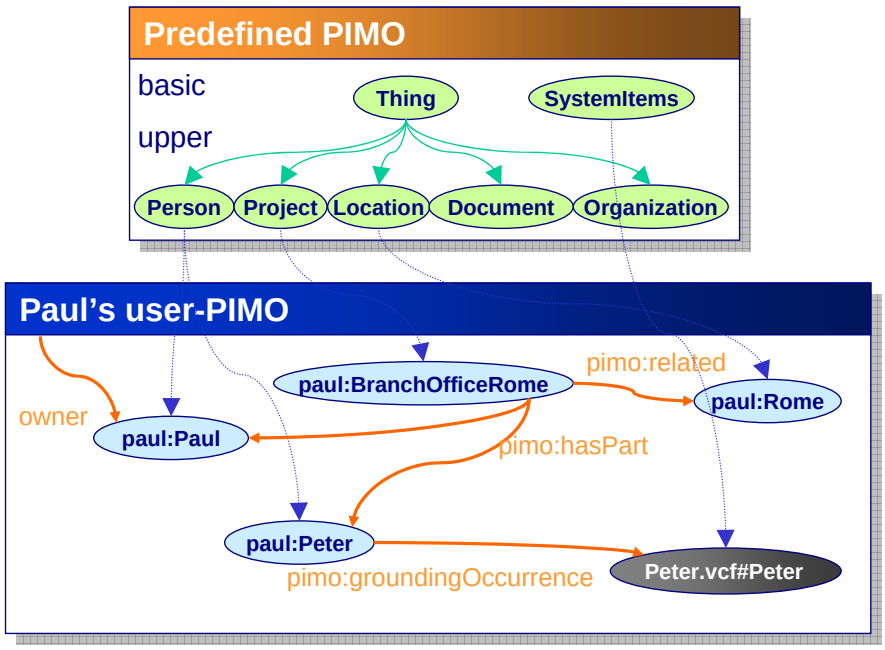


Figure 5.3.: Entities in a user's PIMO

Figure 5.3 shows an overview of the modelling concepts. Let us start with the user, as the model is based on the assumption that information is interpreted by a human being. The Person `Paul` is presented as instance of class `Person`, see Section 5.4.3 for the creation of the user and his user-PIMO.

Following are the entities modelled by the user, which are *Things*. In Figure 5.3, the co-worker `Peter` is represented as a `Person`, see Section 5.4.4. Also the project `BranchOfficeInRome` is represented as instance of `Project`.

There are multiple ways to identify things and to reference to information elements where Things occur, see Section 5.4.6. In the figure a relation to an address book entry is given, a `grounding occurrence`. The classes shown in the upper part of the Figure are taken from the upper ontology in PIMO, which is explained in Section 5.4.12. Each Thing can be described with generic relations, such as the `part Of` relation (see Section 5.4.14) or user-created properties (see Section 5.4.16). In the Figure, the project is related to people who are taking part in the project, `Paul` and `Peter`, and to the related city `Rome`.

5.4.1. Markup of Examples

For convenience and readability, this thesis uses an abbreviated form to represent URI-References. A name of the form `prefix:suffix` should be interpreted as a URI-Reference consisting of the URI-Reference associated with the prefix concatenated with the suffix.

RDF graphs are written in N3/Turtle syntax. Examples serialized as RDF appear in this typesetting:

```
paul:Paul a pimo:Person;
 pimo:isDefinedBy paul:PIMO;
 rdfs:label "Paul".
```

5.4.2. PIMO ontology and namespaces

The PIMO ontology in the latest version [8] has the following namespace:

```
Namespace:
http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#
```

Throughout this document these ontologies and namespaces are used, also indicating their respective versions PIMO is building on:

```
@prefix rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:
<http://www.w3.org/2000/01/rdf-schema#>.
@prefix nrl:
<http://www.semanticdesktop.org/ontologies/2007/08/15/nrl#>.
@prefix nao:
<http://www.semanticdesktop.org/ontologies/2007/08/15/nao#>.
@prefix pimo:
<http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#>.
@prefix ncal:
<http://ont.semanticdesktop.org/ontologies/2007/04/02/ncal#>.
@prefix nco:
<http://ont.semanticdesktop.org/ontologies/2007/03/22/nco#>.
@prefix nfo:
<http://ont.semanticdesktop.org/ontologies/2007/03/22/nfo#>.
@prefix nie:
<http://www.semanticdesktop.org/ontologies/2007/01/19/nie#>.
```

[8] Previous versions are described in [Sau06] but differ only marginally.

```
@prefix nmo:
<http://www.semanticdesktop.org/ontologies/2007/03/22/nmo#>.

@prefix paul:
<gnowsis://paul@example.com/resources/pimo/> .
```

5.4.3. The User and His Individual PIMO

Users are represented as instances of the class `pimo:Person`. For each instance, a new URI is generated and a few key facts are represented to identify the user.

As a prerequisite, each user has a personal namespace. Often these are XML namespaces using the HTTP URI scheme, but any RDF namespace can be used. The example used in this document is `paul:`.

First, the user is represented. Paul is an instance of `pimo:Person`. Additionally, his e-mail address is added. The e-mail address is modelled by reusing the NEPOMUK contact ontology, NCO. In NCO, contact information connected to people is modelled as resource. For the sake of simplicity, we used the URL `mailto:paul@example.com` as identifier for this `nco:EmailAddress` resource.

```
paul:Paul a pimo:Person;
 rdfs:label "Paul";
 nco:hasEmailAddress mailto:paul@example.com.

mailto:paul@example.com a nco:EmailAddress;
 nco:contactMediumComment "work";
 nco:emailAddress "paul@example.com".
```

The second entity that needs to be represented is the *Personal Information Model of the User*. It is connected to the user via the `pimo:metaOwner` relation. For Paul this is:

```
paul:PIMO a pimo:PersonalInformationModel;
  pimo:metaOwner paul:Paul.
```

We further call an individual PIMO instantiated for an individual a **user-PIMO**. Paul's user-PIMO is paul:PIMO. As an abbreviation, it is also correct to write "Paul's PIMO" instead of "Paul's user-PIMO".

5.4.4. Things

The PIMO ontology defines the basic class *Thing* for mental concepts. Every information element encountered in knowledge work by a user is represented as a Thing. A thing is a unique representation of an entity of the real world within one user-PIMO. On the Personal Knowledge

Workspace of a user, a real world entity can be represented in multiple data sources. For example, the person "Peter" may be author of an e-mail, described in an address book entry, and stored in a accounting tool, all part of the workspace of "Paul". In PIMO, one `pimo:Thing` is created as an anchor linking to these multiple representations, such as shown in Figure 5.4.

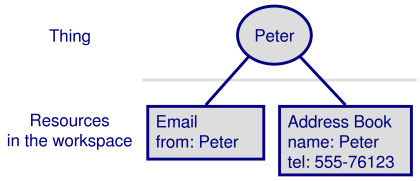


Figure 5.4.: Thing and Resources

An application handling a resource in the workspace has to be aware that there may be a Thing representing the resource. For example, Paul's e-mail client can examine the sender of an e-mail (Peter) and search for the `pimo:Thing` that represents Peter uniquely. Once the right Thing is found by the application, more information about Peter can be discovered.

To be adequate, a PIMO of a user should contain all nameable entities known to the user, but to be efficient, this representation should be restricted to the minimal data needed. Identification is part of this minimal data, and `nao:identifier` provides the property for it.

5.4.5. Connecting Things to the User's PIMO

It is important to know which resources (primarily Things, but also Classes and Properties) were created by the user and modelled in her or his PIMO. For this, the `pimo:isDefinedBy` property is used.

Continuing the example above, this property connects the Person to the PIMO in which it is defined. This is mandatory for every defined Thing and allows applications to identify which elements are part of a user-PIMO and which not .

```
paul:Paul pimo:isDefinedBy paul:PIMO.
```

A `isDefinedBy` property is also defined in RDFS, where resources can be connected to

their defining ontologies, and is also discussed in the light of the OWL standard [9]. The semantics for *isDefinedBy* in PIMO is based on these, with the extension that it is a required property.

5.4.6. Identification of Things

A thing should be represented and identified once, but can manifest itself in multiple elements. For example, the person "Peter" is represented once as an instance of the class `pimo:Person`, and then linked to documents or other things that mention this person.

```
# The canonical Peter
paul:Peter a pimo:Person;
 pimo:isDefinedBy paul:PIMO;
 nco:hasEmailAddress <mailto:Peter@example.com>.

# An e-mail in which Peter #2 occurs
<imap://paul@example.com/INBOX/1> a nmo:Mail;
 nmo:from <imap://paul@example.com/INBOX/1#from>.

# Peter #2, the email sender
<imap://paul@example.com/INBOX/1#from> a nco:Contact;
 nco:hasEmailAddress <mailto:Peter@example.com>.
<mailto:Peter@example.com> a nmo:EmailAddress;
 nco:emailAddress "Peter@example.com".

# Peter #3, as address book contact
<file://home/paul/Peter.vcf#Peter> a nco:PersonContact;
 nco:nameFamily "Benjohn";
 nco:nameGiven "Peter";
 nco:hasEmailAddress <mailto:Peter@example.com>;
 nco:photo <http://www.example.com/people/Peter/photo.jpg>.
```

In this example, we have seen that the Person Peter appears three times in this knowledge workspace. Once, as the canonical instance of `pimo:Person`. Second, as sender of an e-mail and third as entry in an address book. Only one instance is the `pimo:Thing` representation of Peter: `paul:Peter`. The others are representations of the same entity.

For Things, URI identifiers are needed. These should be generated using the namespace of the user. Although they can be randomly generated, we recommend to include the label in the URI for readability. When two things have the same label but are different entities, a random element can be added to the second URI. An URI for Paul could be:

[9] `http://www.w3.org/2007/OWL/wiki/Syntax#Declarations_and_Structural_Consistency`

```
paul:Paul.
```

To work effectively, PIMO is based on the Unique Name Assumption (UNA). Two individuals with different URIs are different individuals. This is common in desktop applications and intuitive to grasp for users, for example files with different names are different. It is different from the OWL ontology language where duplicate entries are common and the Unique Name Assumption (UNA) is not used. The difference is based on the fact that the PIMO is designed for personal systems, where an application has access to the complete model and can avoid duplicates before creating them.

Thus, creating a new thing is always connected to beforehand examining if a thing with a similar name, type, or other identifying properties already exists. Duplication should be avoided.

Things can either be created by the user manually or automatically by analysing existing native resources. When creating a new Thing, identifying properties should be set to avoid duplication and distinguish it from existing ones. Existing identification schemes should be reused for this purpose (for example the e-mail addresses to identify people or ISBN numbers for books) by representing them with nao:identifier and its sub-properties. If an identifier is found as metadata of a native resource (usually a nie:InformationElement), the identifier must be copied to the Thing (manually or by inference). This allows others to match and identify the correct Thing when encountering the next information element.

```
# Copy all identifiers you can find about the Thing.
paul:Peter a pimo:Person;
 pimo:identifier "Peter@example.com".
```

Grounding Occurrence The relation `pimo:groundingOccurrence` is used to link a thing to an `nie:InformationElement` that has this thing as primary topic. For example, the grounding for a person could be the entry in the address book describing the person. A Thing represents the mental concept, the `pimo:groundingOccurrence` links to existing Information Elements that are handled by existing applications. This is a key for reusing the features of these applications. Multiple values are allowed, this reflects the fact that the same thing can be represented in multiple applications, and dependent on the work context, the user may want to open a different application.

```
# Link to Peter #3 from example above.
paul:DirkHagemann a pimo:Person;
 pimo:groundingOccurrence <file://home/paul/Peter.vcf#Peter>.
```

Occurrence The relation `pimo:occurrence` connects a `pimo:Thing` with representations of the same real world entity that is part of the user's knowledge workspace. For example, if the person Peter appears as sender of an e-mail, then the sender is an *occurrence* of Peter. Not the e-mail as such is the occurrence, but the sender within. Occurrences of a Thing can be found by searching for entities with the same identifying properties.

```
# Link to Peter #2 from example above,
# he occurs as sender of an e-mail
paul:DirkHagemann a pimo:Person;
 pimo:occurrence <imap://paul@example.com/INBOX/1#from>.
```

Referencing Occurrence A Referencing Occurrence is an indirect approach to identification. Annotating a thing with an information element as *referencing occurrence* states that the information element contains a description of the thing. Its primary topic must be the thing. The thing is indirectly identified by the element, when two things in different models share the same information element as referencing occurrence, they may be equal and could be matched. The following description is an adaption of XTM's subject indicators [SP01, Rat03]. The referencing occurrence is a kind of proxy for the Thing. Examples of referencing occurrences are:

```
paul:Peter a pimo:Person;
 pimo:referencingOccurrence
   <http://www.example.com/people/Peter>.

paul:ExampleInc a pimo:Organization;
 pimo:referencingOccurrence
   <http://www.example.com/>,
   <http://en.wikipedia.org/wiki/Example.com>.
```

It should contain a human readable documentation describing the concept. The resource could be a document, ontology, video, audio, anything able to describe to a human what the concept is about. The resource is a reference to the concept of the thing. A good example for a referencing occurrence is a wikipedia article.

A referencing occurrence describes the concept with the purpose of being widely used by ontologies. Consequently, it is important the the document describes exactly what concept it is about and what not. Even if the author works as accurately as possible, different people will never interpret a referencing occurrence 100% the same way. However, the concept of referencing occurrences is worth using it, because it allows a shallow match of heterogenous information models, and because there is finally no alternative to it.

Other Representation The pimo:hasOtherRepresentation relation is used to connect pimo:Things with other representations of the same thing in other Semantic Web ontologies. This can be the case with shared ontologies, such as company white page systems or Semantic Social Networking websites.

The knowledge modelled should be compatible with the ontologies used by the user. An example for such other representation is:[10]

[10]Using the URI scheme of the ECS University in our example domain. `http://id.ecs.soton.ac.uk/docs/`

```
paul:Peter a pimo:Person;
 pimo:hasOtherRepresentation
   <http://id.example.com/person/1650>.
```

Another example would be the city of Rome where Paul wants to travel to, linked to the DBPedia entry about it:

```
paul:Rome a pimo:City;
  pimo:isDefinedBy paul:PIMO;
  nao:prefLabel "Rome";
  nao:personalIdentifier "Rome";
  pimo:hasOtherRepresentation
    <http://dbpedia.org/resource/Rome>;
  geo:lat "41.9";
  geo:long "12.5".
```

The relation can be used both to identify things by their other representations, as to fetch more data. In this example, the latitude and longitude are actually superfluous data, they can be retrieved any time via HTTP from the other representation in DBPedia. Assuming Peter also represents Rome in his PIMO, but independent from Paul, but linking to the same DBPedia entry, algorithmically matching their different representations is straightforward.

Other Conceptualization To map user-generated classes to classes defined in other ontologies, the `pimo:hasOtherConceptualization` relation connects classes defined in a user's PIMO with classes defined in domain ontologies.

Implementations can use the `pimo:hasOtherConceptualization` to allow the user and algorithms to map user–specific classes to classes defined in other ontologies, without implying that there is a subclass relationship.

5.4.7. A Complete Example

A complete example for all different identification properties can now be build from above annotations.

For Paul, his co-worker Peter is identified and linked to occurrences like this:

```
# The canonical pimo:Person Peter,
# a pimo:Thing from Paul's PIMO
paul:Peter a pimo:Person;
 pimo:isDefinedBy paul:PIMO;
 nao:prefLabel 'Peter';
 nao:identifier "Peter@example.com";
 pimo:occurrence
   <imap://paul@example.com/INBOX/1#from>;
```

```
 pimo:groundingOccurrence
   <file://home/paul/Peter.vcf#Peter>;
 pimo:referencingOccurrence
   <http://www.example.com/people/Peter>;
 pimo:hasOtherRepresentation
   <http://id.example.com/person/1650>.


# An e-mail in which Peter #2 occurs
<imap://paul@example.com/INBOX/1> a nmo:Mail;
 nmo:from
   <imap://paul@example.com/INBOX/1#from>.

# Peter #2, as email sender
<imap://paul@example.com/INBOX/1#from> a nco:Contact;
 nco:hasEmailAddress
   <mailto:Peter@example.com>.

<mailto:Peter@example.com> a nmo:EmailAddress;
 nco:emailAddress "Peter@example.com".

# Peter #3, as address book contact
<file://home/paul/Peter.vcf#Peter> a nco:PersonContact;
 nco:nameFamily "Hagemann";
 nco:nameGiven "Peter";
 nco:hasEmailAddress <mailto:Peter@example.com>;
 nco:photo <http://www.example.com/people/Peter/photo.jpg>.
```

This allows implementations to:

- identify the Thing when found occurring in documents,
- open a grounding occurrence to see the Thing within an existing desktop application (i.e. the address book entry for a person),
- match this Thing with other representations via the same referencing occurrence,
- use the other representation from the companies white pages to show additional data about the thing.

The `pimo:occurrence` link is the generic basis, `pimo:groundingOccurrence` and `pimo:hasOtherRepresentation` are sub–properties of it. This data should be generated automated and unsupervised. Adding identifying properties to a Thing helps to find more occurrences and therefore more information about it.

5.4.8. Labelling and Naming Things

To label things, we recommend the NEPOMUK Annotation Ontology (NAO) vocabulary. It defines properties for a *preferred label*, *multiple alternative labels*, and a *personal identifier*.

nao:prefLabel *A preferred label for a Thing*. This property **must** be annotated to every instance of Thing. It is used by applications to render the Thing as text and should be user readable. There must only be one prefLabel per Thing (mincardinality and maxcardinality should be one) [11].

nao:personalIdentifier *Defines a unique personal label for a Thing.* The label must be unique within the scope of a user. In PIMO, the personalIdentifier should also be the same value as the prefLabel value, to show the user the same value in most cases. It is a good practice to establish personalIdentifier additionally to prefLabel, as they are unique in applications such as Tagging. Another typical application is a wiki name, or uniquely identifying things within free-text.

nao:altLabel : *An alternative label alongside the preferred label for a Thing.* These are alternative writings, translations, common spelling mistakes that are used for the Thing. Implementations can use these labels to find Things when the user enters a text in a search box or when analysing free text.

In combination, these labelling techniques allow applications to clearly label things in user interfaces but also to lookup for Things based on alternative names. For our example, these are:

```
paul:Peter a pimo:Person;
# a nickname for Peter
 nao:altLabel "Pete";
# a common misspelling
 nao:altLabel "Petr";
# the personal identifier
 nao:personalIdentifier "Peter".
```

Additionally, visual cues (icons, images, thumbnails) can be attached by using NAO symbol relations:

- `nao:hasSymbol`
- `nao:prefSymbol`
- `nao:altSymbol`

[11] These restrictions are not explicitly noted in the RDF description of the property as NRL does not support property restrictions for classes.

Also, **ratings of things** can be added with `nao:numericRating`. For numericRating, the range of values must be within [0..1] (inclusive). We restrict this further in more detail than defined in NAO. Applications may partition the values into discrete ratings (such as 0.2, 0.4, 0.6, 0.8, 1.0 to represent the semantics of "5 star ratings").

5.4.9. Modelling Time

In PIMO, no special treatment of time is modelled. We are aware that representing points in time, durations, and other periods of time is an important aspect of ontologies. Due to the complex nature of time, we recommend to use the XML Schema Datatypes to represent time. There, ISO 8601 is recommended. Timezones must be handled according to this standard, encoded inside the literal value. [12]

To represent periods of time, the class `pimo:PeriodOfTime` is reserved, which defines begin and end times for durations. For durations that last a number of days or months, we recommend to use the standardized XML datatypes:

- `xs:dayTimeDuration` for durations measured in days, hours, and minutes.
- `xs:yearMonthDuration` for durations measured in months and years

There have been issues with other notations of duration and therefore the W3C Semantic Web Best Practices and Deployment Group published a note[13] to restrict durations to these values. The XS namespace is `http://www.w3.org/2001/XMLSchema`.

Periods of time can also be represented using subclasses of the abstract class `pimo:ProcessConcept` which represents lasting processes such as events or projects.

5.4.10. Representing Modification and Change Dates

The change and creation dates of things can be important metadata for personal information management applications. The time-line of recent changes is an important cue for users to retrieve documents, many applications offer the feature to show recent changes or filter by them. Consequently, it has to be straightforward, simple, and fast to query for the modification dates. The NAO properties `nao:created`, `nao:modified`, and `nao:lastModified` are used to track the change dates of Things. Creation and modification allow only value, modification multiple dates.

Example:

[12] For a detailed representation of time events, refer to the NIE documentation, where timezones are discussed (`http://www.semanticdesktop.org/ontologies/2007/04/02/ncal/#sec-tzd`). NIE represents time using the NcalDateTime class and its properties date, dateTime, ncalTimezone. Timezones are represented using a Timezone class, that is inspired by RFC 2445.

[13] `http://www.w3.org/TR/swbp-xsch-datatypes/#section-duration` Since XPath 2.0 has become a W3C recommendation in January 2007, this note is partly obsoleted.

```
# Represent modification dates of a thing
paul:Peter
  nao:created "2007-10-26T15:23:01";
  nao:modified "2007-10-26T15:23:01";
  nao:modified "2007-10-29T08:04:30";
  nao:lastModified "2007-10-29T08:04:30".
```

These values are intended for resources of type `pimo:Thing`, `pimo:Association`, `rdfs:Class`, and `rdf:Property` when created by the user. Setting them is recommended, but not enforced.

The **semantics** of these dates is that the description of the thing has changed, facts about the Thing have been added, removed, or modified. Facts that imply a change of date are relating properties (`pimo:related`, `pimo:hasTopic`, etc.) and describing properties (name, address, label, etc.) that express information about the Thing. Modification of metadata statements (such as `pimo:definedBy`, `nao:modified`) do not imply a change of dates. As RDF stores a priori do not support automatic tracking of changes, applications have to implement housekeeping of these dates, or use services for tracking.

5.4.11. Setting the Class of a Thing

Things are represented using RDF resources and typed using the `rdf:type` relation. Possible classes are `pimo:Thing` and its subclasses. The PIMO ontology itself defines several subclasses such as `pimo:Person` or `pimo:Organization`. If these are not specific enough, the user can either create new subclasses manually, or import mid-level ontologies.

It is recommended to only one class for a Thing. The wish to add multiple classes is often an indication that some classes can be better modelled using relations. Superclasses are inferred implicitly, and are not affected by this recommendation.

5.4.12. The PIMO-Upper Ontology

The PIMO ontology contains an *upper ontology* for basic concepts in Personal Information Management (PIM): *Person, Location, Event, Organization, Topic, Document, Time*. They are modelled to answer basic questions about a thing:

- Who is associated? Person
- Where is this? Location
- When is it? Time
- What is it about? Topic

The classes defined in this upper ontology are intended to serve as integration point for PIM applications. In the broader perspective of the Semantic Desktop, they can serve as upper classes for many ontologies.

5.4.13. Classes in PIMO-Upper

The classes have been defined based on related ontologies, a user study, and several software prototypes that have been evaluated.

Thing The root class of the upper ontology. Every entity that can be in the attention of the user is a Thing.

Collection A collection of Things, independent of their class. The items in the collection share a common property. Several usability studies showed that collections are important for PIM.

Group A group of Persons. They are connected to each other by sharing a common attribute, for example they all belong to the same organization or have a common interest.

Location A physical location. Subclasses are modelled for the most common locations humans work in: Building, City, Country, Room, State. This selection is intended to be applicable cross-cultural and cross-domain. City is a prototype that can be further refined for villages, etc.

LogicalMediaType MediaConcepts are logical media types (e.g., a book, a contract, a promotional video, a todo list). The user can create new logical media types dependent on their domain: a salesman will need MarketingFlyer, Offer, Invoice while a student might create Report, Thesis and Homework.

Organization An administrative and functional structure (as a business or a political party).

Person Represents a person. Either living, dead, real or imaginary. In this regards, similar to `foaf:Person`[14].

ProcessConcept Concepts that relate to a series of actions or operations conducing to an end. Subclasses are defined for Event, SocialEvent, Meeting, Project, and Task.

Topic A topic is the subject of a discussion or document. Topics are distinguished from Things in their taxonomic nature, examples are scientific areas.

These classes are intentionally kept very generic, more specialized ontologies should be used for certain domains of application, and the user can create custom classes.

5.4.14. Generic Properties in PIMO-Upper

The PIMO-upper ontology contains basic relations between Things and a few core attributes for identifying things (described above 5.4.6). These are:

[14] See the FOAF specification `http://xmlns.com/foaf/spec/#term_Person`

pimo:related is the most generic relation, giving no further indication how Things may be related. Related is a `nrl:SymmetricProperty`.

pimo:hasPart and `pimo:partOf` model partitive relations. They are inverse. Neither is transitive, because part-of relations used for modelling in the domain of Personal Information Management are vague due to the many contexts of interpretation (a hotel may be part of a trip plan, a trip plan part of a project, but this does not indicate the hotel to be part of the project).

pimo:hasTopic and `pimo:isTopic` connect a thing of interest with a thing reflecting about it. For example, a meeting can have a project as a topic, or a meeting has a document as a topic, when the goal of the meeting is to discuss the document. After the meeting, the meeting minutes are a new thing having the meeting as a topic. This is not restricted to meetings but also an organization or a person can have a certain technology as a topic to express that they are working on the topic. The relation is not transitive, not symmetric. It is not asymmetric because a document A may have document B as topic, and B also A.

For these generic relations, specialised sup-properties defined when used on specific classes in the PIMO upper ontology.

5.4.15. Refined properties in PIMO-Upper

Additional to above relations, semantically interesting relations between PIMO upper classes are modelled. Especially those which can be used as symmetric or transitive relations for inference.

pimo:narrower and `pimo:broader` relate Topics to each other. As Topics are an important mean to organize document collections based on a taxonomy, these two predicates are defined. They are inverse of each other and transitive.

pimo:hasOrganizationMember and `pimo:isOrganizationMemeberOf` are relations connecting a Person to an Organization.

pimo:hasLocation and [`pimo:isLocationOf`] relate a locatable Thing with its Location. Locatable is an abstract subclass of Thing.

pimo:containsLocation and `pimo:locatedWithin` relate two locations within each other. Note that for geographic locations representing a physical space, inclusion is transitive.

5.4.16. Creating Personalized Classes and Properties

The predefined classes and properties are intended as a generic basis to be extended. The user can always create new classes and property types, or existing ontologies can be imported. New classes are modelled as with RDFS ontologies, with the additional requirements that:

- The superclass has to be `pimo:Thing` or a subclass.
- The class has to be labelled with `nao:prefLabel`.
- The class has to be related to the user-PIMO with `pimo:isDefinedBy`.

The same with custom properties *with things as range*:

- The super-property of a custom property *should* be `pimo:related, pimo:hasTopic, pimo:isTopicOf, pimo:hasPart, pimo:partOf`.
- A property with the class Thing as a range *must* define an inverse property.
- The property has to be labelled with `nao:prefLabel`.
- The property has to be related to the user-PIMO with `pimo:isDefinedBy`.

Inverse properties define the semantic meaning in both ways, which is required for user interfaces showing relations. Custom properties that have *a literal as range* or a type as a range *that is not a thing* are not required to define an inverse. The ranges of user-generated properties should be Thing, a subclass of it, or a literal.

5.4.17. Collections of Things

In Personal Information Management, grouping multiple Things into one collection is a crucial feature. Today's hierarchical file systems are a good example: a folder can be created to contain multiple elements. Later, actions on this folder, such as compressing it, or deleting it are supported. The generic *has Part* relation provides the semantics of putting a Thing into another Thing. For usability reasons, we also provide a class `pimo:Collection` to be used for generic collections of multiple items.

Applications that want to present the complex possibilities of PIMO in a simpler way can offer collections. First, an instance of the class `pimo:Collection` is created. Then, elements are added to the collection using the `pimo:hasPart` relation. A typical application of collections is the list of "Favourites" containing recently used and important resources.

Collections are unordered, the ordering of items inside the collection can be done using alphabetical order, time, geographic location (if they are locatable), or type.

5.4.18. Modeling Associations and Roles in PIMO

Often there is a need to add meta-data about a relation, for example the date of creation of a relation. In RDF, this is typically done using reification, and then adding meta-data about the reified Statement using an instance of the class `rdf:Statement`. A problem with reification is that when using the generic class `rdf:Statement` to represent it, there are no guidelines which properties are now suitable to annotate the statement. More precise sub-classes of Statement would solve this. Another problem is that *n-ary* relations cannot be expressed with triple statements.

In PIMO, **Associations** are used to add metadata about relations and to create n-ary relations. They are entities representing the relation of multiple Things with each other. Each Thing part of an Association is related to the association using the `pimo:associationMember` property or more precise sub-properties of it.

As an example, the fact that Paul attended a meeting can be expressed using the `pimo:Attendee` role.

```
paul:AttendsInitialMeetinginRome a pimo:Attendee;
  pimo:attendingMeeting paul:InitialMeetinginRome;
  pimo:roleHolder paul:Paul.
```

Here, the class `pimo:Attendee` is a sub-class of `pimo:Association` and represents the association as such ("this is an association between a person and a meeting"). The two relations used are sub-properties of `pimo:associationMember` and identify the two Things to relate, the specific relations determine the role taken by each Thing. New sub-classes of association can be created when needed, also new sub-properties of `pimo:associationMember` for more specific roles.

Associations are elements of a user's PIMO and *must* be connected to the user's PIMO with a `pimo:isDefinedBy` relation. Modification dates are to be handled the same way as with Things (see Section 5.4.10).

5.4.19. Integrating Facts about Things

When an InformationElement is the occurrence of a Thing, then the facts of both can be integrated for this user. This is true for `pimo:occurrences`, `pimo:groundingOccurrence` and pimo:hasOtherRepresentation relations. To get a coherent and meaningful view, the class of the `InformationElement` (or related resource) has to be a subclass of the Thing's class, or they are the same.

PIMO inference can be used to **superimpose** statements from occurrences as if they were facts stated about the Thing. Rules for superimposing data can be expressed as SPARQL construct queries. In our prototype implementation, the rules were implemented in program code.

pimo:InferOccurrences: a view that infers occurrences based on nie:identifiers and pimo:referencingOccurrence annotations.

pimo:GroundingClosure: a view that adds statements about the grounding Occurrences and hasOtherRepresentation to a Thing.

pimo:OccurrenceClosure: a view that adds statements about all occurrences to a Thing.

pimo:FullPimoView: a supergraph of all above.

By providing these graphs, we let the user and software agent decide if the full closure is needed at all times. When no closure is needed, the plain NRL data graphs can be used as-is.

To answer complex queries like "Which e-mails were sent to me by attendees of meetings that I have today", the full closure is a good choice.

The ability to superimpose data using inference limits the data needed in a PIMO to a necessary minimum: only the identification properties are mandatory, the grounding and the `pimo:hasOtherRepresentation` properties superimpose existing data. The user should only add information that was not expressed before.

Using above example, an integrated view of Paul on Peter is the following, assuming full closure:

```
# The canonical Peter
paul:Peter a pimo:Person;
# the second type is also inferred
 a nco:PersonContact;
 pimo:isDefinedBy paul:PIMO;
 nao:prefLabel 'Peter';
 nao:identifier "Peter@example.com";
 pimo:occurrence <imap://paul@example.com/INBOX/1#from>;
 pimo:groundingOccurrence <file://home/paul/Peter.vcf#Peter>;
 pimo:referencingOccurrence <http://www.example.com/people/Peter>;
 pimo:hasOtherRepresentation <http://id.example.com/person/1650>;
# the inferred facts
 nco:hasEmailAddress <mailto:Peter@example.com>;
 nco:nameFamily "Hagemann";
 nco:nameGiven "Peter";
 nco:photo <http://www.example.com/people/Peter/photo.jpg>.

# E-mail, now pointing to the canonical Peter
<imap://paul@example.com/INBOX/1> a nmo:Mail;
 nmo:from paul:Peter.
```

From the perspective of the user, facts stated about grounding occurrences are correct (they have been taken from existing data managed by the user); as are `pimo:hasOtherRepresentation` (they are from formal ontologies imported and accepted by the user). If this is not the case, the incorrect statements must not be integrated, which is interesting but out of the scope of PIMO for now.

5.5. Bridges to PIMO: Tagging, Blogging, and Wiki

The PIMO model as presented provides a framework to model things and their relationships, including generic properties such as "related" or "has Topic". Files and resources can be annotated using PIMO by seeing them as things. This representation has the technical advantage of being generic and reusable. From the usability point of view, PIMO is new to users, both the data structures and the interaction is not common in today's desktop applications.

But PIMO can be presented using existing interaction metaphors, namely, *tagging* (see 3.1.4), *blogging* (see 3.1.1) and *wikis* (see 3.1.2). I started research on personal semantic wikis and blogs in 2003 [Sau03] and we have continued in this track [KS05]. In the meantime, others have also published semantic wikis. **Tagging, blogging, and wikis are metaphors building a bridge which web 2.0 users can cross to learn PIMO.**

5.5.1. Viewing PIMO as Wiki

Wikis (as described in Section 3.1.2) allow entering of information in a quick and easy way. They can be employed for both collaborative and personal information management. *Semantic wikis* (Section 3.1.3) are used to author semantic content that acts as *semantic glue* relating information present in desktop applications and desktop data sources such as text documents.

The wiki principles are extended for PIMO:

- Each wiki–page is bound to describe one *Thing* from the user's PIMO. Vice–versa, one Thing may be described with one wiki–page.
- The *wiki name* of the page is the same as the *personal identifier* of the Thing.
- Each page is described using one longer string of *wiki markup*.
- Within the markup, one page can refer to other Things using their personal identifier.
- Hyperlinks can be used to refer to other documents from the web by their *URI*.
- A *semantic wiki markup* can be used to annotate Things within wiki page content.

For example, in the scenario of creating a branch-office in Rome, `Peter` may use the personal semantic wiki to create notes about meetings, involved persons, the project, or the city of Rome. He can take notes during the meeting "`KickoffMeetingRome`" where Peter meets his co-workers `Tim` and `Paul`. They talk about a `BusinessPlanRomeBranch` which is needed to calculate the costs and projected revenue at the new office.

Peter could now write a wiki note with the following text (bold words are links to other wiki pages and are at the same time Things from his user-PIMO):

> Title: KickoffMeetingRome.
> Text: **Tim** noted that he is working on the **BusinessPlanRomeBranch** and has problems matching the rent fees with the projected increase in customer base. **Paul** offered to help here based on his previous experience at the **Tarsus** and **Ephesus** offices. I reported about my positive phone call with local contractors.

Summary on Personal Semantic Wiki In such a note, Peter is able to capture the minutes of the meeting. Especially he can use free text to explain the contributions of his co-workers. The personal semantic wiki is a bridge between text notes and formal semantics in

ontologies, the crucial role of the wiki was confirmed in the evaluations, see Sections 8.8.2 and 9.4.

The semantic wiki markup was described in [Sau03, p. 75] refer to this work for a detailed description. The interaction metaphor of semantic wikis has been described by us in [KS05], for an implementation refer to Section 7.2.8.

5.5.2. Viewing PIMO as a Personal Semantic Blog

While the personal semantic wiki is designed to describe the properties of specific Things, the need for a general personal note-keeping tool also exists (for an introduction of blogging, see 3.1.1). There are different activities that support personal information management and can be realized in a personal semantic blog:

- Ad-hoc capturing of short notes to remember facts. During information work, new facts can arise and the need to write them down. Creating a complex document and storing it into a folder is then often too cumbersome and the information is lost. To capture short notes, a personal blog can be used.

- Daily summaries of activities. Some disciplined knowledge workers keep a daily rhythm of planning and reflecting: the daily activities are planned in the morning, after the work-day a short summary is written about the events of the day. Both is recommended in popular time-management literature. The daily summary can be seen as a way to keep a *business diary*.

A problem with conventional tools to achieve these note-taking tasks is that they are not connected to the rest of the information in the user's personal space of information (PSI). With the personal semantic wiki and PIMO, the user can refer to elements from his or her own PIMO when capturing notes. Technically, this is already implemented in the personal semantic wiki (previous Section).

Summary of Personal Semantic Blog The PIMO model in combination with the personal semantic wiki can be used as personal semantic blog. By using this feature, users are able to file *short notes* during the work-day and *a personal business diary* at the end of the day. The text in these notes links to PIMO Things, therefore weaving the personal semantic web tighter by adding contextual information about Things.

5.5.3. Viewing PIMO as Tagging System

In the last Section the metaphors of wiki and blogging were mapped to PIMO, in this section a mapping of the *tagging* model to PIMO will be done. As described in the introduction (Section3.1.4), *tagging* means to attach words to resources, like adding "Rome" to the website `www.comune.roma.it`. Tags are a flat approach to categorization, each resource can have

one or many tags, and the tags themselves have no meaning besides there name and the connected resources.

The model of tagging is adapted to be used in combination with the PIMO model:

- For tagging within PIMO, each resource has to be represented as Thing first. If a resource is to be tagged and no Thing exists, a new Thing is created and related to the resource using the `pimo:groundingOccurrence` relation.
- A *Thing* is subject of one or many *annotations*.
- A *tag* is shown to the user as a keyword, internally it is represented as a *Thing* and the keyword is stored using the *personal identifier* property of the Thing.
- The *user* is represented as an entity, but is not needed within the PIMO model as everything is indirectly connected to the single user of the system already.
- An *annotation* is a set of *Thing*, *tag* (another Thing), and *date*. The annotation is stored as a statement of *Thing–pimo:hasTopic–tag*. The date is stored as an annotation of the reified statement.
- Multiple annotations build the tagging system.

For the user, we can then present every *Thing* with a unique personal identifier as a possible *Tag* for other things. The semantic meaning of a tagging annotation is that one *Thing* has another *Thing* as topic. In **addition to tagging systems**, PIMO offers the power of ontologies:

- Tags can have alternative spellings using the `nao:altLabel` annotations.
- Tags are Things, hence they have a class and are part of a taxonomical hierarchy and not just a flat list.
- Tags are Things, hence they can be related to other Tags.
- Tags are Wiki pages, hence they can be described using semantic wiki text, in the text other things can be mentioned to create semantic links between things.

Summary of Personal Semantic Tagging The PIMO model in combination with tagging and a personal semantic wiki interweaves the features of tagging, wikis, and blogs. At the same time, some common caveats of tagging systems are solved: as all tags are Things and are part of a formal ontology, structure and alternative spellings can be introduced for tags. The tagging metaphor is later used in the tagging plugins for existing applications, see Section 6.4.6.

5.5.4. Summary: A bridge to web 2.0 users

In the last sections we have shown a core asset of the PIMO approach for the Semantic Desktop. **Although the underlying model is a semantic web ontology, the user interaction can be simplified to existing web 2.0 features: wikis, blogging, and tagging**. Through the merging of ontologies, wikis, blogging, and tagging into one coherent model, a bridge was built allowing users trained in today's existing technologies to use the Semantic Desktop. By allowing users to tag their files, e-mails, and visited web-pages, they can indirectly fill their Personal Information Model. By keeping short notes in the semantic wiki, they enrich the relations and context information about created things.

5.6. Integrating Domain and Mid–Level Ontologies

Mappings between ontologies were either realized by using subclass/sub-property relations to map classes or by using the custom property `pimo:hasOtherRepresentation` to express the fact that one instance $A1$ of ontology $O1$ is represented in another ontology $O2$ in the instance $A2$. $A1$ `pimo:hasOtherRepresentation` $A2$ would be the according triple. These mappings were primarily used to match instances created by the individual user in his individual model to instances in domain ontologies. For example, the user creates an instance for the project "Rome Project" and later connects it to the instance in the organizational repository representing the same project. The services for peer-to-peer ontology alignment and organizational-memory (OM) wide ontology management are described in [vEK04].

5.7. Boundaries of the PIMO

In this Section the boundaries and limitations of the PIMO model are discussed. First, the scientific boundaries of the model are described by giving examples where it fails and where it is beneficial. This also includes implementation issues. PIMO *coverage boundaries* (5.7.2) are limitations to the scope of modelling: not everything in the world must be modelled, but enough to capture daily knowledge work.

5.7.1. Scientific Boundaries

The PIMO model is designed to capture named entities, their semantic meaning, relations to other entities, and semantic wiki text describing entities.

This modelling is scientifically founded in related work, which was listed in Section 5.2.1. The problem of the scientific approach is the evaluation of ontologies in the field. Up to date, there are only scarce evaluations about how the related work was used in practice. We miss literature about the acceptance of ontological modelling (classes, instances, relations, rules) by knowledge workers in daily information work. Which classes are practically used? Which relations are understood by users?

In PIM activities, the information structures are often by far more complex that the PIMO model. For example, the relation between a Person and a Project is not a simple "has Part" relation. In large projects, the **simple relation of a Person and a Project** actually includes much more information:

- The *Person*
- The *Project*
- The *Organization* of the person (is the Person representing a certain Organization within the Project)
- The *Role* of the person for this Project (programmer, director, legal, quality assurance, etc)
- The beginning and end date of the relation to the project
- etc.

This example is not artificial, but occurred in the evaluations reported later in this thesis. Often, not the full level of detail is needed, so a simple way to express the relation would be useful. Representing this relation thoroughly in PIMO would mean to model it as association (Section 5.4.18), which can get cumbersome. For personal information management (PIM), it may be enough to write the role, dates, and organization facts as textual descriptions into the semantic wiki.

To sum this problem up: **to express complex relations, we are not able to clearly recommend a best practice**. All three ways (simple relation, association, wiki text) are possible and field studies about their advantages and disadvantages are missing. Hence PIMO fails in capturing the exact relations between Things, users tend to use simple relations for modelling and keep the real semantic relation in their mental model (see Section9.4 about the evaluation result).

From the **implementation aspects** we realized all modelling constructs in our reference implementation. Building a user-interface for associations was not achieved in the first prototypes, but associations can be stored in the database and others have shown how to create such user interfaces[15]. An implementation and realization problem was the *distinction between Things and Information Elements*. An Information Element can either be modelled as resource which is "tagged" as being about Things or as a distinguished Thing which is related to other Things. In the related Topic Maps approach (as introduced in Section 3.4.3), documents are not modelled as Things. This separates the concept world from the document world and is typically understood well by users. Documents may address, deal with, or refer to concepts that are explicitly defined, which may expressed by Topic Maps and was adopted for PIMO.

Our own users experienced a problem when they wanted to further annotate documents, a seamless transformation from an Information Element into a Thing would be a desired feature.

[15]For example, the Freebase service has a complex association editor that could be ported to PIMO.`http://www.freebase.com`

We were not able to find literature about this transformation process, documents have a dual nature both as "entities which can be tagged" and "entities that can be tags for other entities". In later prototypes, the upper class `pimo:Document` was used to represent all documents as PIMO Things, but this approach flooded the PIMO model with numerous entities and decreased the precision of the model[16].

5.7.2. Coverage boundaries of a user's PIMO

A problem with the personalized view of the world arises when facing information overflow — the world is fascinating without end. We may be tempted to include everything into a personal information model (the *MyLifeBits* project aims at recording and storing everything a user sees and hears [GBL+02]). But not everything can and must be stored, as storage space is limited and information overflow can be avoided.

When ontologies are defined as conceptualizations of a domain of interest, then the domain of interest for PIMO are *resources that the user is aware of*. Elements that need to be represented in a PIMO can then be defined as *elements which are consciously recognized during information work*. In the following, we use the terms "awareness", "attention", and "consciously" synonymous for the same principle.

Let us call this statement the "*Awareness rule of thumb*": *Resources that receive attention of the user are to be represented as things in the user's PIMO.* Based on it, further rules can be deducted that guide software engineering when coping with this model. Firstly, the time aspect is extended: resources that *were in the attention in the past*, and resources that *may be of interest in the future*. Every resource that is annotated by the user as a Thing in the present tense is also stored and available for retrieval, becoming an item of the past. Based on mid-level ontologies (domain ontologies) and interests of the user, the system can guess which resources will be interesting in the future and annotate them as things.

Secondly, the level of awareness can vary. Over all resources available in the users knowledge workspace, each will receive a different level of attention. We did not develop a formal representation of these attention levels. Various relations between a user and a resource exist: "read the resource", "annotate the resource", "understand the resource", "use the resource as input", "receive or send the resource", "create the resource", "store the resource", "resource is part of the knowledge workspace and mid-level ontologies but is never seen".

Applied examples can be a business plan that Paul has created himself, which receives high attention throughout a longer time and is without question a Thing in the user's PIMO, and on the other side a company guideline describing how to create business-plans, which was never seen by Paul but was all the time available in his knowledge workspace. A system based on PIMO could represent the business plan as Thing, and be able to search for the guideline resource and suggest it as a possible Thing that may need attention in the future. An example for awareness levels are all the files that make up the operating system (libraries, configuration, system utilities): in

[16] Representing all documents as Things can result in millions of Things in a user's PIMO, all e-mails, all files on the user's hard-disk, all visited websites. This contradicts our approach to limit the boundaries of a PIMO as described in the next Section5.7.2.

normal operation these are never in the attention of the user. They are not part of the PIMO and would only distract. But once the operating system stops working as expected, and a certain log message becomes important and a configuration file needs to be modified, these elements become part of the PIMO. They become things and related information from internet FAQs for troubleshooting can be annotated to these files.

The ongoing *MyMory*[17] project aims at measuring the attention a user gives to resources based on user observation and eye-tracking. The results of this project will contribute to the understanding of user attention.

Semantically and technically speaking, things and resources differ:

- A Thing has a unique, automatically generated unique URI within the user's namespace. The URI of a Thing identifies the conceptual aspect of the Thing.
- A resource is identified with a URI that is determined by the storage location and the protocol how to access it. A resource URI can change when systems change or documents are moved.
- Statements about a resource are both about the physical and the conceptual level. A document may have an author but also captures concepts, such as a topic, an event, or projects on a conceptual level. It is also described on the physical representation layer. A data object can have a byte-size, an encoding scheme, or a MP3 file has a bitrate. To describe the physical aspects, there are precise vocabularies such as EXIF for pictures, for conceptual annotations there is the Dublin-Core terms for documents.
- Annotations about things express facts about the concept represented by the Thing. This extends the annotation possibilities with PIMO mid-level and upper-level ontologies, that allow the user to express that a document is actually a contract, and that it is related to other things on the conceptual layer.

Feedback gathered in the user study showed that from the semantic perspective of the user, a Thing is annotated. The user does not distinguish between annotations on the resource level (*bitrate of MP3*) and annotations on the upper- and mid-layer of things (I like the music, the project is managed by Paul). In the evaluated prototypes, the user interface showed an artificial distinction between resources and things, but users wanted to express their thoughts using the expressiveness of mid-level ontologies and of data ontologies in combination.

5.8. Summary on PIMO

In this chapter we have presented the *Personal Information Model (PIMO)* ontology framework. It is used conceptually to represent the mental models of the user, practically to annotate resources and express knowledge needed in daily information work. We based our research on

[17] `http://www.dfki.uni-kl.de/mymory/`

existing approaches (primarily related work on PIM ontologies, Semantic Desktop and on existing ontology languages) but consciously chose to create a new vocabulary for our purposes. The conceptual layers of the framework were explained and practical implications and use of the model was shown for the scenario of the example knowledge worker "Paul".

We have experienced that our previous designs of PIMO were not accepted by users nor developers if not done properly (precise representation, easy adoption, easy to understand by users, extensibility, interoperability, reuse of existing ontologies, data integration). Although some modelling decisions remain a compromise and may not be shared by the reader, we have designed this ontology with great effort according to our best knowledge and known sources.

PIMO in use

The PIMO approach was deployed and used in several research projects, it was the basis for the EPOS project, it is also the basis for data representation in the *gnowsis* project, which was described in [SGK+06], and the NEPOMUK project [GHM+07].

Norberto Fernandez created an approach to populate a PIMO while the user is doing search tasks. The user interface of his SQAPS search engine automatically creates PIMO concepts in the background, annotating them with Wikipedia pages [FGSSB06] as referencing resources.

The SeMouse project used our gnowsis implementation and the PIMO model as an extension of their work [IAD06].

Vinh Tuan Thai et al. have developed a graphical document browser to explore large document spaces based on the PIMO ontology and published their work at ESWC 2006 [THD08].

On Mobile

Woerndl and Woerl ported the PIMO idea to mobile devices, studying how ubiquitous access to personal structures can support mobile users [WW08].

In the rest of this thesis, the PIMO approach is the underlying information model for personal information management (PIM).

CHAPTER 6
A Semantic Desktop Architecture for Personal Information Management

The personal semantic web consists of strings of various texture. In the previous section, the *Personal Information Model* PIMO was introduced, giving a fabric of ontologies that can, in principle, be used to annotate files and resources of interest to the user. Also, the PIMO forms a categorization scheme consisting of things, that can be presented as complex RDFS classes and instances or simpler as taxonomy of topics or just tags.

To support *Personal Information Management (PIM)* in its full meaning (Section 2) — *the management of data in the personal knowledge space as performed by the owning individual* — we developed a Semantic Desktop software architecture that extends existing operating systems with novel data formats, novel services, and novel applications. This architecture is realizing a way to use PIMO from multiple applications accessed by one user. *Creating this architecture provides an empiric evaluation of the applicability of the PIMO model.*

PIM is not limited to a single application or user interface, as suggested in Haystack [QHK03], or to integrating applications in a one-to-one manner [Boa04]. Instead, we propose that existing applications have to be adapted to be part of the Semantic Desktop. A big task at hand, but using conventional engineering, it can be split into manageable parts. First we introduce the *aims* that PIM can target on the Semantic Desktop (which form requirements and influence architecture), then the needed *functionalities are split into multiple services*. The services are invisible software components, *applications* use them and build the interface of the user to her or his personal semantic web.

A description of the implementation and the internal architecture of the services is found in the following Chapter 7 and is downloadable from the project website[1]. Some of the described services and applications have been implemented only in gnowsis 0.8.3, others in gnowsis 0.9.3. All were also taken as input and further improved in the architecture of the NEPOMUK EU project. For the following text, I picked the *simplest and complete* versions of each service from the different implementations, to support understanding the functionalities they offer. For the evaluation, different services were evaluated separately. Independent of implementation, the use for *Personal Information Management* stays the same.

[1] `http://gnowsis.opendfki.de`

6.1. Aims and Requirements

The **requirements** on a Semantic Desktop system depend on the approach, as we can see in the related work in Section 4. My chosen requirements are an extension of [Sau03] and [DAB+02]. Details about each follow after the overview.

6.1.1. Functional Requirements

The following requirements affect the functionality and the interfaces of the services.

- **Single user**: the services are designed to work on the data of a single user.
- **One PIMO model for one user**: all applications should be able to access and modify the user's PIMO.
- **Support PIM activities**: typical PIM activities (filing, finding, maintenance) as described in PIM literature must be supported.
- **Multiple views on the same data**: data can be shown and navigated in different ways.
- **Support for Ontologies, Taxonomies, Wikis, and Tagging**: both complex and simple interaction must be possible on the same Things modelled in the user's PIMO.
- **Automated and manual annotations**: annotations can originate from automatic algorithms, manual entering, and semi-automatic combinations of both.

Single user

The Semantic Desktop, as understood by myself, first aims to enhance the existing way of managing information for a single user.

Based on this requirement, the architecture can be simplified. The RDF data representing the PIMO of the user can be stored in a single database. The contents of this database belongs to the single user and therefore no access restrictions are needed.

Similar simplifications can be applied for other services: the interfaces of the services should assume to be called by the process of the "current user" and hide identification, authentification, and multi-user support from the programmer. This requirement is realistic: when the "current user" is the owner of the calling thread, no identification is needed.

One PIMO model for one user

Users need categorization and grouping functionality, but **each application should not re-create its own scheme**. With the PIMO we have a model how to represent this in a unified way, but now the PIMO categorization has to be integrated into various applications. Applications should not create their own categorization scheme but reuse the PIMO categorization created by other applications.

Related to this requirement is the restriction to one user: one user's PIMO can not be shared completely with another user's PIMO. As each user has his own subjective view on the world, a separate PIMO is needed for each user. Of course, this does not restrict communication and exchange between users, but a synchronisation process between users is then needed. Multi-user support and social exchange is out of scope of this thesis.

Support PIM activities

In the field of Personal Information Management (PIM), there is no canonical final truth on what activities are considered to support PIM and how to realize them best. But the results from many field studies show typical activities. For the author, the most helpful collections of PIM literature are the ongoing PIM workshops held at various conferences [TJB06, JB05].

There, three types of activities are identified [JB05, p10]:

- Keeping activities affect the input of information into a Personal Space of Information (PSI)
- Finding/re-finding activities affect the output of information from a PSI.
- "M-level activities" (e.g., "m" for "mapping" or for "maintenance and organization") affect the storage of information within the PSI.

On the Semantic Desktop, support for filing information as resources, annotation of these resources, and re-finding of resources are primary concerns. M-level activities include annotation, data enrichment, and semantic expression of knowledge, which are considered secondary for this work. M-level activities can also include personal time-management and task-management, which we only cover in part in as they have been addressed elsewhere in depth [HMBR05].

An additional important factor to support PIM is that *users need manual categorization schemes to support creative thinking and reflection*. The act of filing information does not only change the filed information, but also involves a cognitive decision-making process, changing the mental model of the user [BN95]. As [KJ06] pointed out:
"There are more basis reasons to organize information — we understand the information better. People often have, and complain about having, several distant organizations of information — usually folder hierarchies Integration means providing at least an option to bring these organizations together. Some people may still find it useful to have distinct organizations for email, e-documents and web references. But this would be a choice freely made, not a separation imposed by supporting applications."

Multiple views and applications work on the same data

There are various approaches to visualization and viewing information [GBL+02, FG96, QHK03]. In the related work we find two-dimensional visualization on planes, alphabetic views, taxonomy and hierarchical views, time-based views, context-based views, three-dimensional

ways [EAD+06], etc. For different tasks, different views are needed, so the user should have the freedom of choice between multiple views on the same information. On the other side, the view implementations need to have access to all information underneath in an integrated way.

There is no requirement that all these interaction metaphors have to be realized as an integrated, single application, such as Haystack [QHK03]. As a requirement, the services must be accessible from multiple programming languages and from multiple applications running in parallel. There can be multiple annotation editors, browsers, viewers, search interfaces.

Existing applications used in PIM should rather be extended with plugins rather than replacing them with "new semantic web applications". This is also a fact well-known in usability: user's are reluctant to change their way of working. Let them keep their applications where users are trained and comfortable than force a switch to new applications.

Support for Ontologies, Taxonomies, Wikis, and Tagging

The services must reflect the heterogeneous views of PIMO (as defined in Section 5.5) and allow users to access the same data through different interfaces. A complex service interface is needed to interact with the data as an ontology or taxonomy, simpler service interfaces are needed for interaction on the level of wikis or tagging. All services must work on the same data model.

Automated and manual annotations

The PIMO model should allow storing the annotation of Things both by automatic algorithms and by manual operations. In general there can be *four* ways to build the user's PIMO, but the field is open for many more (or less).

First, the user himself can manipulate his PIMO. For this, a **user interface for PIMO editing** is needed; which can manifest as ontology editor (with a big complexity and not easy to understand), or as simple tagging plugins that allow to annotate resources with keywords.

The personal semantic wiki and blog system is our main input, allowing the user to express knowledge using a semantic wiki syntax. Alternatively, we have created more approaches that allow users to manipulate their PIMO.

Second, **existing text documents can be used as input for natural language processing (NLP)**. Documents form the basis of Personal Information Management today, and information about projects, people, places and more can be extracted from text. The problem is to convert existing data sources to a Semantic-Web data format. In this thesis, the NLP approach was not evaluated, but this requirement is nevertheless important and has been tackled in much more detail by others [Hor06], and the general field of NLP and ontologies [BPM05, BTMC04].

Third, a **combination of user interface and automatic analysis** should be possible. Once the user starts annotating a document in a user-interface, the document should be analysed and compared to the existing PIMO. Suggestions should be made how to annotate the document, and how the contents of the document would change the PIMO further. This requirement can be seen in the *Drop-Box* application (see Section 6.4.5), related work is *ConTag* [ASRB07].

Fourth, the **social environment** of the user may contribute to his or her PIMO. The company a person is employed by—or a project the person is working on—influence the semantic inter-

pretation of information. Assuming the company structure or the projects of the user are already represented in a formal, machine readable way using ontologies, how can these **domain-specific ontologies** be imported to augment the user's PIMO?

This will be later addressed by the importing functionality of the *PIMO Service* 6.2.3.

6.1.2. Non-functional requirements

The non-functional requirements are derived from the functional requirements. They serve as design guidelines to realize a cost-effective implementation of the prototype, and also to ease adoption of the architecture in real-world applications.

System level, not application level Other Semantic Desktop architectures are built as monolithic applications [QHK03, MMY04, DH05, CPG05, TMN06, XC05]. Many features should be integrated into one system, but the question of integration of the application into operating system processes is not tackled. I envision the Semantic Desktop as an extension to current operating system functions (for example, additional to "store a file", an operation to "annotate a file" must exist).

It must be possible to implement all services as daemon processes that can be contacted by multiple applications in parallel.

Standards based The services must work on RDF, semantic web, and other existing standards to reduce the integration and training costs. Where protocols are needed for inter-process communication, existing standards should be used.

Modularized The architecture is based on a *separation of user interface applications and services*. The services should have well-defined interfaces, and have no hidden side-effects. When possible multiple small services are preferred amongst one bigger service.

Performance and Stability Implementations must perform their operations within a user's reaction time (i.e. a search result must return faster than a second), take not too much CPU and memory cost, and be stable enough to work for a prototype implementation. The service architecture must be implementable.

6.1.3. Design Approach

Three areas in the realization process are influenced by the requirements: **ontologies, services, and applications**. The overall architecture of the system consists of these three parts and they interlink—if the ontology changes, the services have to be adapted, and the applications showing the data provided by the services. In reverse, new needs in the user interface may affect services and ontologies.

The presented services were built in an iterative process of incremental prototyping and incremental improvements.

6.2. Services

In the following sections, we describe the core services that were used to realized the gnowsis Semantic Desktop prototype, as evaluated. Each service runs as part of the local desktop semantic web server, offering functionalities to other services or to end-user applications. The services are intended as **personal services**, available on the desktop to support the individual user.

In Figure 6.1 an overview of the architecture is given. It can be compared to a **tree**.

- The personal RDF store builds the base of the tree, where all information of the system is kept (Section 6.2.1). The roots of the tree are invisible services growing underneath the surface, working to manage the data.

- Like roots extracting "nutrients" from existing native data sources, the *Data Wrapper* services connect to existing applications and convert data to RDF (see Section 6.2.2). Other services analyse this data, enrich it, and align it to form a coherent PIMO for the user.

- The *PIMO* service implements basic methods to manipulate a user's PIMO (Section 6.2.3). Semantic wiki text can be managed using the *Personal Wiki Service* (Section 6.2.7). A simple way to interact with PIMO concepts is provided by the *Tagging Service* (Section 6.2.6).

- The *Categorization and Resource Similarity Service* can suggest tags and related documents given one document as input (Section 6.2.5). Searching on the PIMO data and extracted data in an integrated way is implemented in the *Search Service* (Section 6.2.4).

- The current work context of the user is computed based on observed user operations in the *User Work Context Service* (Section 6.2.8).

- Finally, a *User Interface Service* provides backend methods to trigger the user interface (Section 6.2.9). For example this service can be called when an application inspects a document to show the PIMO annotations of the document – a generic browser will be called by the backend.

The trunk of the tree is the local Semantic Desktop server, exposing the services. Above the soil of data, visible to the user, are various applications which build upon the invisible services. The semantic applications are the fruits offered to the user for PIM.

For each service, we will describe in short what functionalities it offers, what interface exist to communicate with it, and possible usage scenarios of the service. Parts of this are published by us in [Sau03, SGK+06, SS05a] and similar approaches exist (amongst them [Gel05]). Parts of the service descriptions were published in my diploma thesis [Sau03], as part of the gnowsis project [SGK+06] and in the NEPOMUK project deliverable 2.1 [MS06].

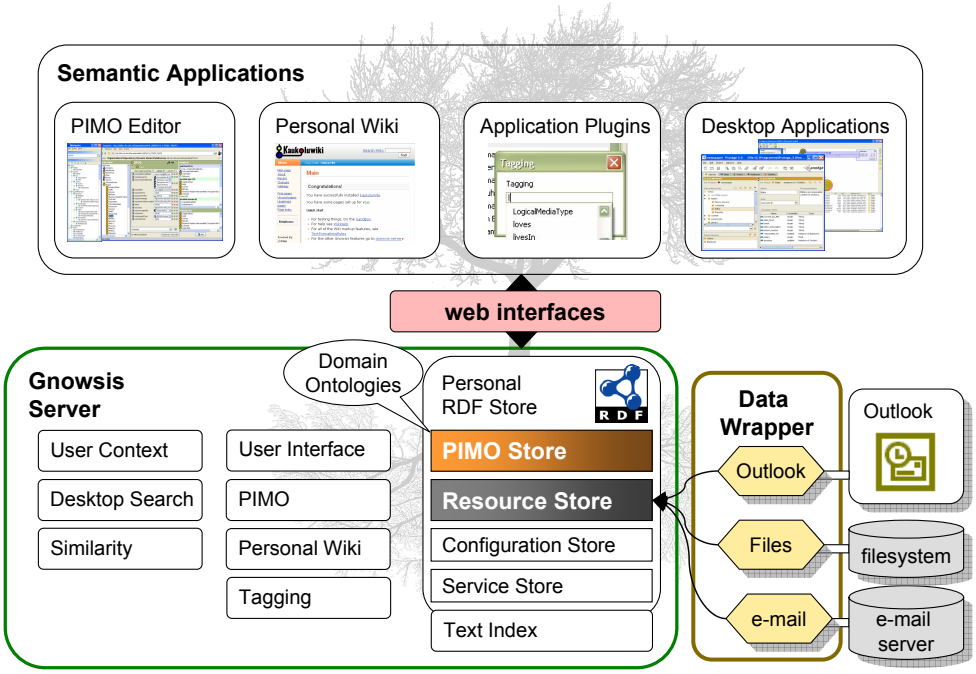


Figure 6.1.: The Gnowsis Architecture

6.2.1. Personal RDF Store and Indexing and Annotations

Personal Store

All desktop applications can use the personal **RDF Store** to share metadata about resources, and through this sharing it is possible to integrate them on the level of data and ontologies. Information about the person `Paul` edited by application A can be changed and annotated by application B, and vice versa. If one of the applications can express information in a format not readable for the other, the RDF model will allow both facets to co-exist without disturbing each other. For example, if A can work with telephone numbers but B does not, then the telephone numbers added by the first will not disturb correct functioning of the second. The mechanism behind is the concept of extensible ontologies, a basic concept of RDF that we will not explain here. Both ontologies and data are stored in the `RDF Store` service.

RDF Store Interface

The interface[2] conforms to well-known RDF APIs. For gnowsis, we implemented the interfaces both of Jena and Sesame2, the most popular RDF APIs for Java. On an abstract level, the interfaces consist of the following methods.

[2]Interface: `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnowsis/repository/Storage.html`

`addStatement()`	add one RDF statement
`removeStatement()`	remove one RDF statement
`find()`	find statements that match the passed pattern
`querySelect()`	run a SPARQL select query and return the result table in the standardized SPARQL Query Results XML Format[3] (or, optionally, as an object).
`queryConstruct()`	run a SPARQL construct query and return the result graph as RDF (serialized as text or as objects).

More complex methods exist to add graphs, remove graphs, and query graphs, but they can be expressed using above atomic methods. Note that the RDF store must implement a **context-aware quad store**. That is a typical feature found in modern RDF APIs, for each subject-predicate-object statement, and additional fourth value (therefore *quad*) can be stored. Often referred to as *context* of the statement, in can be used for various functionalities [CBHS05], in Section 7.2.2 our exact usage is described.

The query languages supported by the RDF Store are SERQL, as it is supported by the underlying RDF store, and SPARQL, the W3C standard for RDF query languages [Pe05].

Fulltext index

Being the primary storage medium for PIM data, a full-text indexing capability was a needed functionality. From the related work we are aware that searching in the text of resources is as important as searching by structure or categorization. We combined both functionalities in what we call a *LuceneSail*, a combination of full text indexing and RDF store in one. All statements are automatically indexed when adding them (to be more precise: the literal object values of statements are indexed). Querying can be done using a fulltext search alone, or a combined search of fulltext and RDF using an enhanced query engine. We extended the SERQL query engine in our prototype.

Querying the fulltext index is then realized using *"magic"* predicates in SERQL queries. For example:

```
SELECT X FROM
{X} <http://something/matches> {Y},
{Y} <http://something/query> {"java"}
```

this will return any X that has some field indexed with the word "Java". More details about this functionality is described in the implementation Section 7.2.1 below.

Functionality

The main functionality of the RDF store is integration of data from various sources, and the possibility to annotate the data with cues for retrieval or information management. On a fully semantic-desktop enabled system, all structured data from the personal knowledge workspace would be available in the store, or kept in RDF-enabled systems.

6.2.2. Data Wrapper

There is need of a service that acts like a data integration hub, working to integrate data from existing applications or external data sources. The service realising this is `DataWrapper`. As

the RDF database starts empty, but a user may already have much information created using existing applications, this existing information should be reused. Local files, e-mails, address book entries and other data sources are read from their native file formats, converted to RDF and indexed in the RDF Store. This again lowers the integration effort for application developers: instead of reading several file formats to integrate a calendar and a task list, the DataWrapper does this and provides it as RDF data. Developers can then access the RDF Store to query for information.

Access to data

Shown in Figure 6.2, the approach to data integration in pre-Semantic Desktop times can be compared to our proposed architecture. Traditionally (the left side), accessing data through an application programmer interface (API) involved multiple steps. First developers had to learn different **protocols** and interfaces to access data from other applications. On the Desktop, these are typically COM and *ActiveX* on Windows or DBUS on Linux. SQL, http, CORBA, and LDAP are typical examples of client-server protocols. Once the connection to another application is established, there are different **data formats** that can be found, and need to be understood to work with the data. There are text formats such as HTML or PDF, and structured formats such as XML, or the data are kept in tables, as in a relational database (DB). To learn what the **semantic meaning** of the data is, the developer had to read software documentations, or, if there are none available, guess the contents.

Using the *DataWrapper* service, access to data is simplified to accessing RDF data. The **interface protocol** is SOAP or SPARQL, the **data** format is RDF and the **semantic** meaning of the data is explained in ontologies. And in RDF, most structured content can be represented. Simplifying the steps needed for data integration is the key benefit of the DataWrapper.

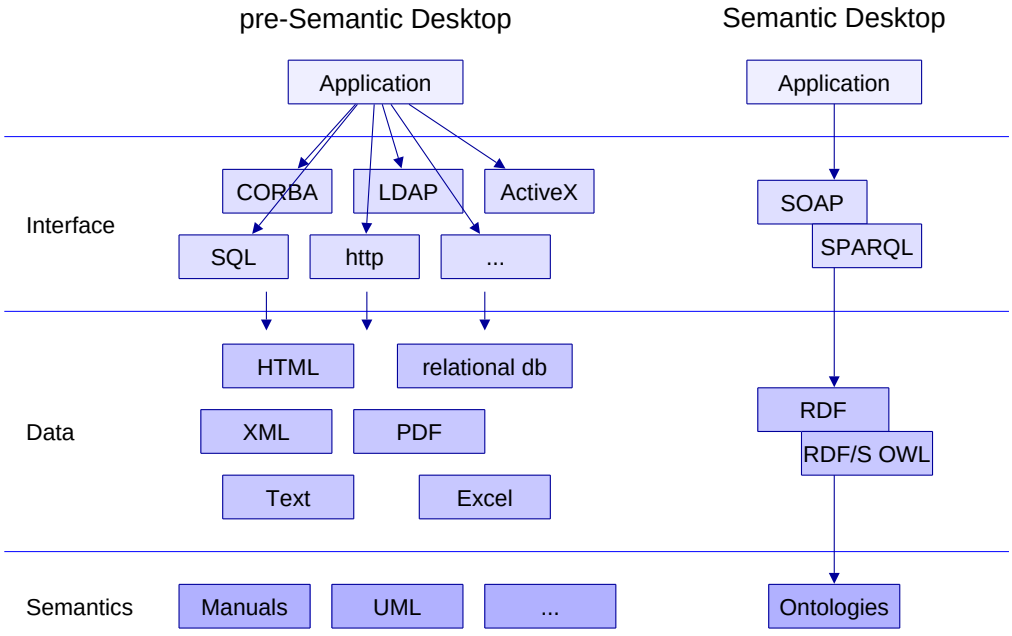


Figure 6.2.: Data access stack

Additionally, it allows opening resources within their native applications. For example, if a semantic application sees that a task is related to a person, and the user wants to open the person for editing, the address book application would be invoked. Existing applications are reused instead of replacing them. The Data Wrapper is conceived as a façade to external native applications.

The service itself is structured into several plugins, which are also exposed in the interface. Figure 6.3 gives an overview of the internal architecture of the Data Wrapper[4]. First are *data sources* which represent data to be integrated, and settings how to access this information (passwords). Second are *crawlers* that access the data inside the data source and iterate through its contents. Part of the contents are binary streams, that are encoded according to MIME types. Third are *extractors* that convert a binary stream into RDF by interpreting the MIME type. The man-in-the-middle is the *crawler handler*, coordinating the process. The handler collects the converted RDF data and stores it into the *Personal RDF Store* service.

Data sources, crawlers, and extractors have clear interfaces and are managed as plugins to the component, making it possible to extend the functionality by adding more implementations. In the following, the functionality of each is explained.

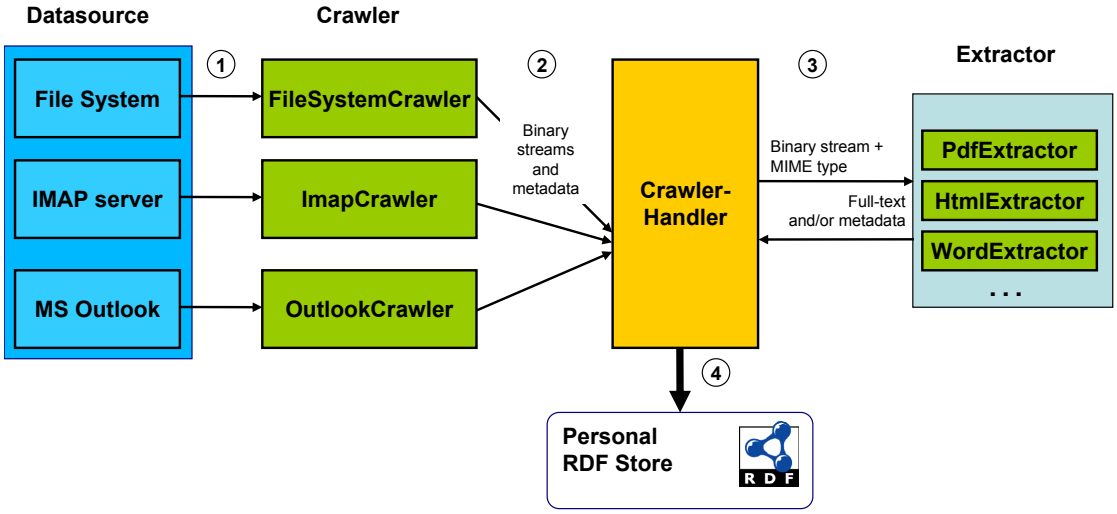


Figure 6.3.: Data Wrapper internal Architecture

Data Source

A **data source** is an entity that represents a store of data. For the data wrapper to work, the type of source, the location of its contents, the user's credentials to access it, and rules defining which content to index are needed. Examples of data sources include file systems, mailboxes, remote servers, relational databases, etc. Data sources can be considered at various levels of granularity, depending on the intended usage. A file system may be treated as a whole (as a set of files, organized in a directory structure). Also, individual files can be a whole data source, for example a calendar application may store all its data into one vCalendar file. Data source configurations are needed to define what of this data should be integrated.

[4]This architecture was developed by myself and Chris Fluit for Aperture, published before in [SGK+06].

Crawler

A **crawler** is a software component that is capable of iterating over data items available in a *data source*[5]. It starts at an entry point set in the data source and touches all resources which should be indexed. As an example, a file crawler would start in one directory and iterate through all files, and then all sub-directories recursively. Crawlers can be implemented for many structured data sources (Microsoft Outlook, e-mail servers, databases, etc.).

Incremental crawling

Incremental crawling is a concept used to describe a feature of a crawler that enables it to keep track of the state of a data source between extractions and report only those data items that have changed.

Extractor

An **extractor** is a software component capable of extracting RDF data from a raw stream of bytes. The input can be taken from various sources, both local (e.g. files on a hard disk) and remote (via HTTP, FTP, SMTP or any other protocol). Depending on the MIME-type of the binary stream, different extractors are needed. In the DataWrapper, extractors for common document formats (office files, PDF, JPG, MP3) should be available.

Example In an example, a **file system crawler** iterates through files in a location defined in a **data source**. One of the files is an PDF text file. Picking the right **PDF extractor**, the crawler will extract the metadata and plaintext of the file as RDF and store it into the RDF store.

Interface to the DataWrapper

The DataWrapper interface has been deliberately designed to be as simple as possible. It provides methods to configure data sources, start the crawling process, to stop it, and to determine if it's being executed at the moment. These methods are enough to integrate existing data into the personal RDF store. Additionally, there are methods to trigger the crawling of *one resource* and to open one resource within its native application (i.e. to open an e-mail within the e-mail client).

DataWrapper interface

The DataWrapperAPI interface[6] contains the following methods (parameters omitted, exact documentation can be found in the implementation):

`addDataSourceConfig()`	configure a new datasource
`updateDataSourceConfig()`	update the configuration of a datasource
`removeDataSource()`	remove a datasource
`start()`	start crawling
`stop()`	stop crawling
`isRunning()`	returns `true` if the service is crawling, `false` otherwise
`getCrawlReport()`	a report on the crawled resources
`accessResource()`	crawl and index one particular resource now, using the configured crawler and extractors.

[5] The term crawling is used as in *web crawling*.

[6] `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnowsis/data/ApertureSynchroniser.html` and `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnowsis/data/ApertureDataSourceRegistry.html`

6.2.3. PIMO Service

The PIMO Service help with the handling of ontologies, classes, and instances stored in the RDF Store.

If the store can be seen as an Excel Spreadsheet, storing data in tables, then this component can be compared to a convenient way of creating new tables and adding rows to it. For example, to capture a list of colleagues, in Excel one would create a new table called "Colleagues" and a row would be added with data for "my colleague Tim". Using the `PIMO Service`, first a class "Colleagues" would be created, possibly a subclass of "Person". Then "Tim" of class "Colleague" would be added.

The data itself is stored using RDF, by the *RDF Store* service. Developers can still create RDF data themselves and add it directly, the PIMO Service provides a convenient way for some often needed functionalities. For example, the creation of unique URIs for new resources is implemented in the `createUri()` method. If the type of the resource is known, the method `createResource(name, type)` can be used. This will create a new URI for the resource and store the type information, then additionally it can store metadata like creation date and creator (the local user).

PIMO Service Interface

These methods are based on underlying ontologies, in the simplest way they reflect the semantic of RDF Schema. Examples for methods from the PIMO Interface [7] are:

`getUserNamespace()`	get the namespace of the user (see Section 5.4.3)
`createUri()`	create an URI for a new resource
`getUserUri()`	get the URI identifying the user (see Section 5.4.3)
`createResource()`	create a new RDF resource, returns its URI
`deleteResource()`	delete a resource and all data stored about it
`createClass()`	create a new class, returns its URI
`deleteClass()`	delete a class, not possible if instances exist
`addOntology()`	add a new ontology to the store

More methods allow to manage **properties**, so altogether the service supports **ontologies, classes, properties, resources**, and **relations**.

Creating new classes, resources, or properties requires the service to generate new URIs. We distinguish between the *personal information model* of the user (a kind of personal ontology) where new URIs are created using the namespace registered at a namespace provider (see Section 7.2.3); and ontologies that are imported from external sources, then the URIs have to use the external namespace.

Social Ontologies

By importing external ontologies, information about the work domain of the user can be added to his PIMO. For example, the company where the user works, colleagues, projects, topics can be modelled in a company ontology. Importing this ontology is realized by the PIMO-service.

[7] `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnowsis/pimo/PimoService.html`

To make the implementation more flexible, rules can be used to define what will data will be added or removed when a method is invoked. Rules can either be expressed using custom Jena[McB01] forward-chaining rules or SPARQL-construct queries[Pe05] (which allows the creation of RDF data).

6.2.4. Search Service

The search service provides methods to do complex searches on the data stored in the RDF Repository. Additional to the functionalities of the store (SPARQL, SERQL, and fulltext search), the search service provides functionalities for ranking results and to infer and return additional result items based on rules.

Search Interface

The interface of the search service[8] is restricted to a single search method, returning the results either as HTML for a user interface or as RDF encoded result. The parameters supported by this method are:

String query a fulltext query expressed using lucene's query language

boolean inference apply rules on the result or not

boolean distributed include data from colleagues in search

String store which repository to search

Passing these parameters, these two methods are available:

`getQueryResultHTML()`	Run the query and return the result as rendered HTML page (parameters are explained above).
`getQueryResultRDF()`	Run the query and return the result as RDF.

The query result was expressed in the *GnoSearch* vocabulary inspired by the Roodolf RDFS model for the Google api[9]. It contains classes to represent the search request, each returned hit, the rank of the hit, and data for visualization (labels, a text-snippet, explanations)[10].

Representing the search results as RDF before rendering them as HTML output allows the service to run personalization rules before rendering the results. Based on horn-clauses this search engine can be extended and adapted. Example rules are:

- If a RDF resource has a literal matching the query, include the respective typed resource in the result list.
- If a document was found, add its concepts to the result list.

[8] `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnowsis/search/GnowsisSearch.html`

[9] `http://nutria.cs.tu-berlin.de/roodolf/rdfs`

[10] For the full vocabulary description, download the RDFS description from the namespace here: `http://www.gnowsis.org/ont/gnoretrieve`.

- If a concept has a special link (e.g., "hasOtherRepresentation"), include the connected resource into the result list.
- If a project has been found, include also its project leader in the result.
- If in the result a resource is found which is an instance of one of the four main PIM classes shown on the result page (Person, Project, Concept, Event14), then list this resource in the respective top-level area.

These rules are expressed using the Jena[McB01] Rule syntax as described in the Jena documentation. We also embedded the possibility to call additional SPARQL[Pe05] queries from within the rule engine, hence a rule can decide to expand the search by invoking another search to the ontologies. Personalized sets of these rules can be used to expand the search results (increasing recall values) or to filter out unwanted results (increasing precision). In the EPOS scenario, rules were used to include defined ontology mappings (hasOtherRepresentation links, see above). This is a shortened version of the rule set used in the evaluation of the system presented in Section 11:

```
# found something?
# -> infer other representations via SPARQL
(?hit retrieve:item ?x) ->
querySparql('CONSTRUCT
 { ?x pimbasic:hasOtherRepresentation ?y }
')

# found a project?
# -> also show members
(?hit retrieve:item ?project),
(?project rdf:type org:Project) ->
querySparql('CONSTRUCT   {
  ?project org:containsMember ?m.
}).
```

6.2.5. Categorization Service and Resource Similarity

When archiving a new resource, the system should recommend Things that can serve as a classification. There are different algorithms possible to find possible Things related to a resource. Resource similarity can be used to compare the semantic description of the new resource with existing, already categorized resources and suggest the categories of the existing resources. Another way is text similarity, comparing the fulltext of the new resource with already stored resource and then suggesting tags that have already been assigned to existing resources.

This service can implement various such algorithms inside, the interface leaves this open. In our prototype, we have used text similarity for categorization suggestions.

Categorization Interface

The interface for categorization [11] consists of one method. In the interface, the term **tag** is

used synonymous for **Thing**. As described in Section 5.5.3, Things can can be interpreted as tags in a tagging system.

`getPossibleTagsFor()`	get possible tags (=Things) for this resource.

Training

The service can be used by any application that needs suggestions to classify and categorize resources. The quality of returned suggestions will increase over time, as *training data* is needed to know how the user interprets the categories. Basically the algorithms work on similarity measures.

6.2.6. Tagging Service

The Semantic Desktop provides multiple ways of viewing and manipulating the same data, this is also reflected in the services. The tagging service allows to annotate things using keywords as tags. The keywords are represented again as things, as described in Section 5.5.3. The service is used to add tags to things, search for possible tags by substring, and suggest possible tags (for this, the categorization service is reused).

Communication with the service is based on the concept of *native resources*, which can be *tagged* using things. Documents such as e-mails, websites, or files can also be tagged. The API is a convenience API, it serves as illustration how concepts of PIMO can be simplified, the methods are:

`addTagToResource()`	add a tag to a resource
`createNewTagForResource()`	create a new tag and annotate the resource with the tag
`getTagsOf()`	get the tags that are already associated to this resource
`getTagsWithName()`	get tag(s) with exactly this name
`getPossibleTagsForTagName()`	find possible tags that contain the passed name

Tagging in use

Based on the tagging API, it is possible to implement various tagging plugins without knowing much about the Semantic Desktop system nor without knowing anything about RDF. This lowers the cost of implementing plugins and applications,

6.2.7. Personal Wiki Service

The personal wiki service provides methods to programmatically interact with the personal semantic wiki. Using wikitext as knowledge markup should not be restricted to the wiki application alone, it is possible to interpret "comment" fields in existing desktop applications as wiki text. Doing this, commenting on a resource can be reused to relate the resource with other things and

[11] The interface is part of the tagging interface: `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnowsis/pimo/TaggingApi.html`

annotate it further. Our interpretation of a semantic wiki also allows to annotate other resources in wiki text. Preliminary the wiki has to be turned from an application to a service.

The interface of the wiki service[12] concentrates on storing wiki text, retrieving wiki text, providing HTTP links to the user interface of a wiki page, and to relate semantic entities (things) with wiki pages through wikinames. The last part is important — every thing has a name which can both be used as a wikiname and as a tag.

`deletePageByUri(String thingUri)`	For the passed URI of a Thing, delete the wiki page.
`getEditUrlForLabel(String thinglabel)`	get the a URI to a web-based user interface to edit the wiki page of the Thing with this name.
`getViewUrlForUri(String thinguri)`	get the URI of a web interface that can be used in a browser to view the wiki page of the Thing with this URI.
`getPageTextByUri(String thingUri)`	Gets a specific version out of the repository.
`putPageTextByUri(String thingUri, String text)`	Attempts to save the page text for the page identified by the URI of this Thing.

For example, the `getViewUrlForUri()` method can be used to get an URL that, if entered in a browser, shows a minimal semantic wiki browser and editor for the Thing identified by this URL. Taking this "view URL", a browser user interface widget (such as Microsoft's Internet Explorer ActiveX control) can be turned into a semantic wiki comment field in many desktop applications. The other methods were used to augment existing user interfaces with a connection to the semantic wiki.

6.2.8. User Work Context

The main challenge for context representation and reuse of context is the definition of a context model ontology for the personal knowledge management domain. In [Sch06] Schwarz explains a pro-active, context-sensitive assistance system to aid the user during her knowledge work, which is mostly about searching, reading, creating, and archiving of documents. This system was built as a research prototype in the EPOS project. Focus was to avoid distracting the user, therefore context gathering is realized by installable user observation plugins for standard applications such as Mozilla Firefox and Thunderbird. This and other context modeling approaches have been discussed in the introduction in section 2.7. The interface of the user work context service[13] provides methods to notify the service about operations of the user, and to query the currently relevant PIMO Things.

[12] `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnogno/api/WikiApi.html`

[13] `http://www.gnowsis.org/statisch/0.8/javadoc/org/gnogno/usercontext/UserContextApi.html`

`addNativeOperation()`	Add an observed native operation of the user.
`getRelevantConcepts()`	Return resources that are relevant to the current work context.

For the Semantic Desktop, Schwarz adapted his approach to use the personal information model as underlying data basis. The modelling was also used for this thesis. Schwarz evaluated the approach in the EPOS project and continued it in the follow up projects MyMory and NEPOMUK. For a detailed description of the representation of context, refer to the deliverable 2.2 of the NEPOMUK project and [Sch06].

6.2.9. User Interface Services

Besides the wiki and opening resources using the DataWrapper, there are other user interface services defined for the Semantic Desktop. To open resources for a user interface, there are several generic methods defined in the browser API[14].

`browse(String URI)`	Show a semantic browser for this resource, analyse the type before and pick a special browser, if registered.

For browsing, different user interfaces are defined to browse classes, properties, and Things.

As a simple way to link resources across applications, we realized the "link" user interface metaphor. Pressing "link" in one application will show a popup-window that shows the selected resource and allows to add more resources to link either via drag-drop or by pressing "link" in more applications. The simple interface[15] of the methods again allowed to reuse it throughout many user interfaces.

`linkResource(String resourceURI)`	Link a resource. Chooses itself if the resource is subject or object of the created statements.
`linkObject(String uri)`	Link this entity as Object part of a triple.
`linkPredicate(String uri)`	Create a link using this predicate.
`linkSubject(String uri)`	Create a link using this resource as subject part.

Also, user interfaces need standardized icons and previews for all kinds of resources across the desktop, these are handled by the IconService[16].

[14] `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnogno/api/Browser.html`

[15] `http://www.gnowsis.org/statisch/0.8/javadoc/org/gnogno/linker/Linker.html`

[16] `http://www.gnowsis.org/statisch/0.9/doc/gnowsis-server/javadoc/org/gnogno/iconservice/GnowsisIconService.html`

`getSmallIconFor(String uri)`	Return the small icon for this resource.
`getSmallIconForType(String typeuri)`	Return the small icon for this RDFS type.
`getThumbnailFor(Resource r, Rectangle maxSize) (String typeUri)`	Get a preview thumbnail for this resource.

Semantic Clipboard

Reif has defined a semantic clipboard API [RMG06] which should also be part of this section (but is not, this task remains for future work).

In combination, the link and browse functionalities allow a very low-level and fast integration of existing applications to the Semantic Desktop. No changes to existing user interfaces are needed besides adding two buttons. A more complex point to build extension is given by the Tagging service. Altogether, the services have been used to create various applications, which is the topic of the next sections.

6.3. Example Usage of the Services

Tag a file

After knowing about the core services, the next questions is how to interact with them to achieve a programming goal. A simple process will be illustrated: **tagging one text file**. In the next section, more complex applications and user interfaces will be presented that work based on the services.

The example to "Annotate a file" is well suited, as it covers different services. Assuming that the user "Paul" wants to annotate file "Business Plan.doc" (further called P), a MS-Word document. The system should propose a few tags based on the text content and then let Paul also add a new tag.

In Figure 6.4 the overall process is shown. First, the user interface calls Aperture Data Wrapper passing the file-URI of P. The method `accessResource(`P`)` is called (1), passing P as a parameter. This will instruct Aperture to analyse the file, identify it as a MS-Word document, invoke the right extractor to transform the binary stream into RDF (2), and store the plaintext and RDF metadata of the file in the Resource Store (3). The call returns to the user interface, which can now ask for recommendations and classification of the document from the Categorization Service. The method `getPossibleTagsFor(`P`)` is called (4), again with the parameter P. The Categorization service uses text similarity methods based on the Lucene Index to find possible tags (5). The list of tags $Possible(T)$ is returned as URIs identifying the PIMO Things in the user's PIMO. The user interface can show these tags and let the user select: $Selected(T)$. For each selected tag T, the Tagging service is called, adding the tags to the document: `addTagToResource(`T`,` P`)` (6). Internally, the relation between the tag (the thing) and the file is stored as a triple in the PIMO store (7). The user can also add a new tag, by entering the tag's name and creating the new tag. As example, Paul enters the name "Rome", the user interface calls Tagging-Service to create the tag and add it to the resource in one operation: `createNewTagForResource(''Rome'',` P`)` (6,7).

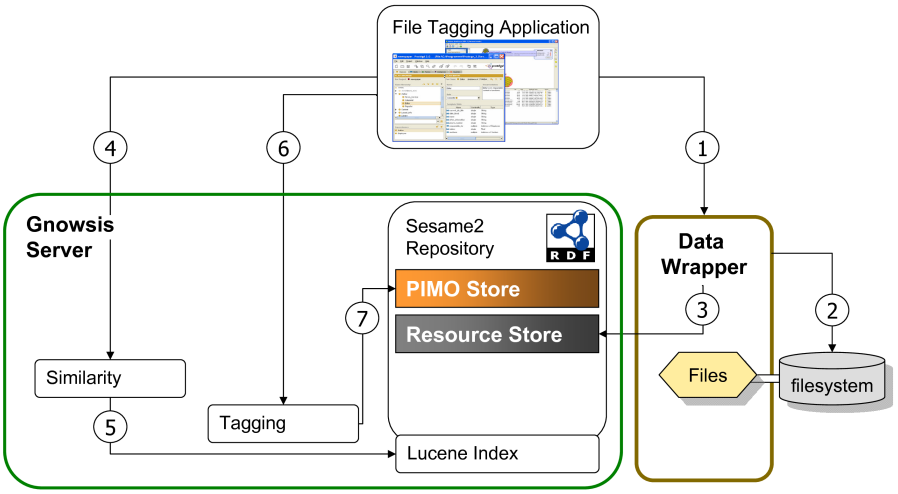


Figure 6.4.: Example: Tagging a File

After these operations, the resource has been tagged with one new tag and several suggested tags, all annotations stored in the PIMO Store.

6.4. Applications for PIM

Given the new services and ontologies on the Semantic Desktop, applications for Personal Information Management can be found. The focus is to allow the user to benefit from her or his personal information model in different tasks.

Our Semantic Desktop reference implementation gnowsis can be seen as prototypical implementation of a Semantic Desktop GUI. It is an interface that allows browsing and editing information. Users can open a desktop application and see the data of their PIMO, manipulating it, editing it, and configuring the Semantic Desktop. The design of this interface has seen many changes in the last years, and based on user evaluations and related work, was continually adapted. In 2005, Dominik Heim analysed the status of the GUI and came to the conclusion that the interface can be split into two main components: a *sidebar* and a *browser*. Several other applications are implemented to support annotation of files, tagging websites, and search.

The applications were implemented by the team of various gnowsis developers, leading

amongst them Dominik Heim for the Swing user interface, Gunnar Aastrand Grimnes for the Web 2.0 visualizations, Malte Kiesel for the personal semantic wiki. Other developers include Florian Mittag, Ralf Biedert, Daniel Burkhart, and Sebastian Weber. I managed the work, designed the architecture, and was lead programmer.

In the next sections we give short descriptions of the applications that proved useful in the evaluation.

6.4.1. Miniquire Sidebar

The sidebar conforms to the *auxiliary posture* user interface metaphor, as described in [CR03], it is a mixture of transient and sovereign posture; meaning the application plays the role of a "silent reporter" that can be quickly moved out of the way when necessary. As shown in figure 6.5 the sidebar contains a search interface on the top and below an overview on the user's PIMO. Called "Miniquire" [17], it allows users to quickly find things inside their PIMO or manipulate them. It contains functionality for adding and deleting (sub-) classes or things as well as more sophisticated options like hiding (e.g. unimportant) things or star them, which means the are marked with a star to ease the retrieval. Furthermore Miniquire provides filters to show hidden things again or limit the personal ontology to show either things or classes or both. The user can even use drag and drop to rearrange items within his PIMO.
Miniquire represents the primary user interface for providing an overview and managing the personal ontology.

6.4.2. PIMO Thing Editor: Unified Annotation of Resources

The *browser* shown in figure 6.6 enables the user to focus on a specific thing to see all relations as well as providing the possibility to edit the relations and metadata of a thing. The center of the user interface is the free-text area where the semantic wiki *kaukolu* (see section 6.4.4) is embedded. On the right side, the important relations of the thing are shown as a list. Clicking on the related things allows navigation to them. The relations can be edited using drag-drop operations, also files and web-links can be added to a thing by drag-drop.

Editing PIMO things using the thing editor allows three functionalities:

- Editing the wiki-text describing the thing.
- Editing relations to other things and files using the relations editor.
- Editing string attributes of the thing (visualized below the wiki-text).

Editing the wiki text opens the Kaukolu wiki editor in a web-browser, see section 6.4.4.

Editing the relations to other things allows adding or removing relations to other things. As standard relations (immediately visible when the editor shows) the editor provides the following PIMO relations:

[17] The name is a pun referencing the work that was before, Tim Berners-Lee's Enquire.

- related
- part Of
- has Part
- has Topic
- is Topic Of

Adding a thing as "part Of" means that the related thing is part of the currently edited thing. New items can be added to the lists using the search boxes to the right or by drag-dropping them from other windows of the user interface. This can be seen as what Rohmer calls "Explicit Semantics" [Roh05], a possibility for the user to write information directly in the semantic network.

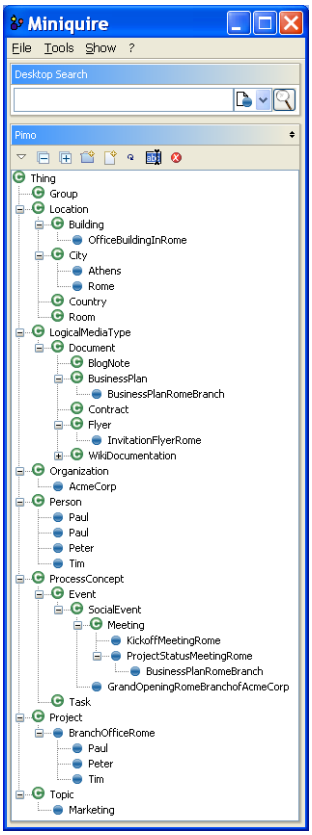


Figure 6.5.: The sidebar user interface "Miniquire".

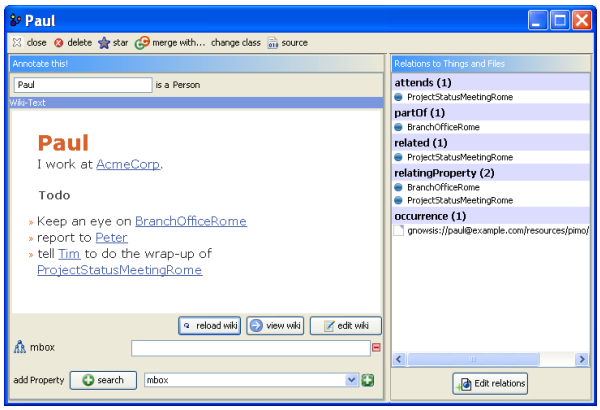


Figure 6.6.: The "ThingEditor" browser and editor.

Additional, some special properties are shown:

occurrence Files or other resources where this thing occurs, meaning resources that describe the thing as topic.

grounding resource Resources that represent the shown thing in native applications.

folders File folders (or e-mail folders) containing resources that have this thing as topic. This information is also used by the Dropbox application.

6.4.3. Various Browsing Interfaces

The various dimensions captured in the PIMO imply various interfaces to interact with them. For the prototype, we have implemented several common Web 2.0 browsing interfaces allowing the user to see the same information from different angles.

Four interfaces were created.

- A geographical map of PIMO locations. It was realized using google maps.
- A timeline view of the events, realized using simile's timeline widget[18]
- A tag cloud of relevant PIMO things.
- A bookmarklet to annotate web pages with PIMO things.

The geographical view, timeline, and tag-cloud are self-explanatory and don't need further explanation given the fact that we use them everyday in web 2.0 applications. The bookmarket is a little different as it is not so common. Bookmarklets are one-line javascript applications that are published on websites as hyperlinks. The javascript application is encoded as a URI using this notation:
`javascript:alert('hello').` They are added to the web-browser by drag-dropping them into the favourites (or "bookmarks", therefore also the name). Usually, they are used to invoke actions based on the currently viewed page, the javascript engine can access the URI of the currently open page.

We have used the technique to realize a browser-button implementing "tag this page in gnowsis". Pressing the button opens a web user interface that allows annotating any website using things from the PIMO. The feature is popular through services like del.icio.us.

Refer to Figures 6.7, 6.10, 6.8, 6.9 for the implemented user interfaces.

6.4.4. Personal Semantic Wiki

Traditional wikis enable people to collaboratively author a set of interlinked texts (*wiki pages*). The idea of semantic wikis is not only to edit texts but author information that can be processed by automatic means. In practice, this means that semantic wikis aim to support advanced queries

[18] `http://simile.mit.edu/timeline/`

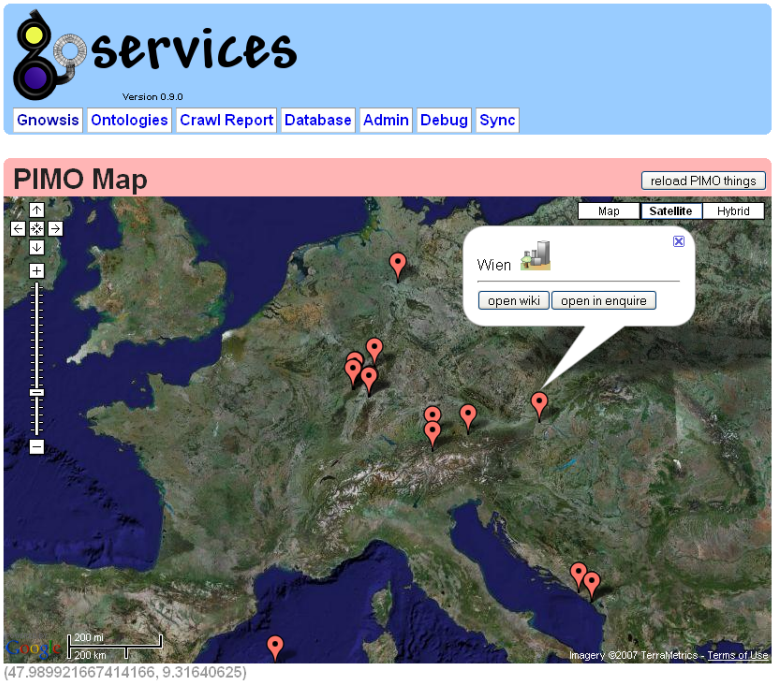


Figure 6.7.: Geographical Map

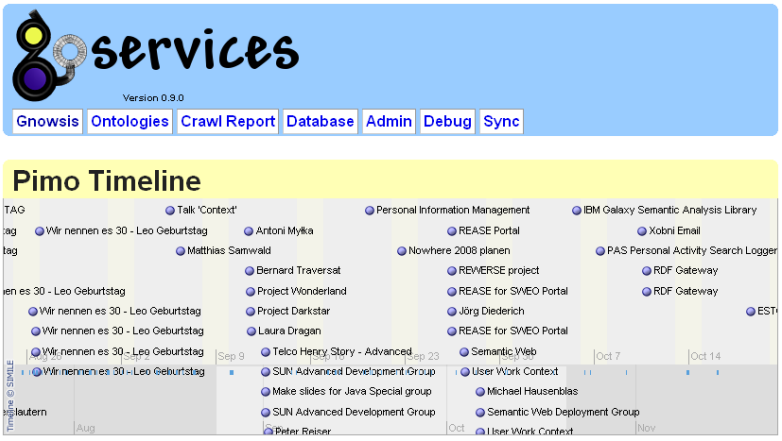


Figure 6.8.: Timeline View

Figure 6.9.: Tagcloud

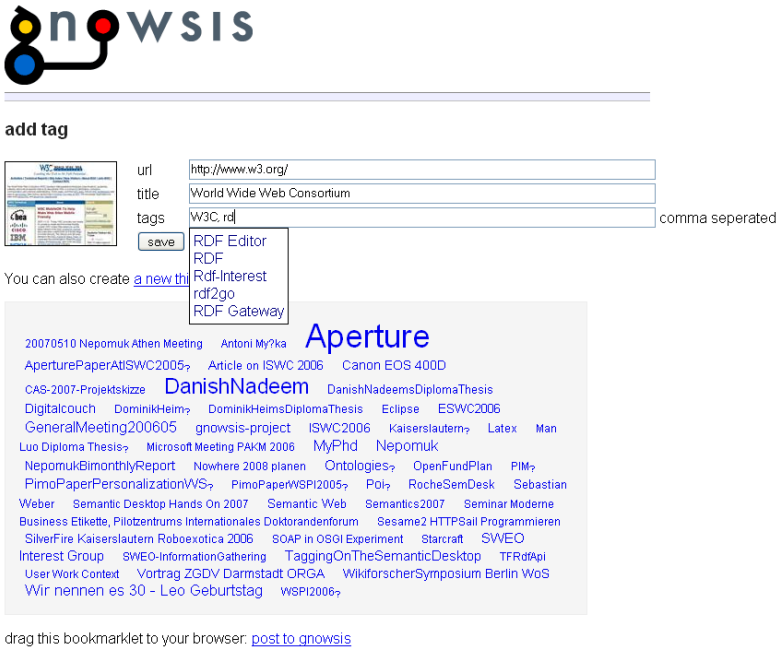


Figure 6.10.: Tagging Bookmarklet

("What semantic web researchers wrote more than 60 papers?") and advanced search ("Search for *ant* as in *software*.").

Gnowsis integrates with the semantic wiki *Kaukolu*[19] [Kie06]. The main idea is that a wiki page can be created for every instance in the user-PIMO ontology. In Paul's PIMO, there would be wiki pages for *Rome, Italy and Paul*. Note that each wiki-page is automatically a *tag*. This means that every gnowsis resource can be browsed in Kaukolu and vice versa. The same is true for relations between resources which can be created either gnowsis or Kaukolu. In gnowsis, relations are created using the standard GUI, while in Kaukolu, relations are written in a plain text syntax that is similar to N3. The user gets supported interactively with an autocompletion feature when entering data. This relieves him from having to know every relation's name or URI. The autocompletion feature bases its suggestions on ontologies stored in the PIMO-storage.The integration of Kaukolu with gnowsis opened up for several interesting features:

- **Browser integration**: With the wiki, it is possible to use the browser as a simple frontend to the gnowsis system. We even plan to move some of gnowsis' more advanced features to the wiki by way of using wiki plugins.
- **Simple data input**: Wikis are a well-known tool for authoring texts without the need to adhere to rigid templates. This can be used in the Semantic Desktop context, too, as with the wiki it is possible to add unstructured information (for which either no schemas exist, which are too costly to formalize, or no benefit in formalization can be thought of) to any desktop resource present in the gnowsis system.
- **Formal data input**: the semantic features add the possibility to author complex relations within the wiki text.

Semantic Wiki Syntax

For these features, a *Semantic Wiki Syntax* was created. This syntax is an extension to existing wiki syntax. In addition to links to other wiki pages, it is possible to add links to RDF resources and PIMO-things from the user's PIMO. Relations between multiple things can be expressed as sentences, descriptions entered on one page can affect things defined elsewhere.

The semantic wiki syntax was first published in [Sau03]. Basically, it allows the user to augment sentences with RDF entities such as:ties such as:

```
Wiki page about ''Rome Project''.
This is about the new offices in [Rome].
[Rome] is a [City]; [part of] [Italy].
[Tim] also [works on] the [Rome Project].
```

Each sentence in the text is parsed and interpreted in a language similar to Tim Berners-Lee's *Notation 3* [BL98]. The first sentence has no simple RDF statements encoded, only a plain PIMO-related link between the *Rome Project* and *Rome* will be created. The second sentence adds the class *City* to the thing *Rome*, and in the sub sentence it is situated as part of *Italy*. The third sentence relates *Tim* to the *Rome Project* using the *works on* relation.

[19] `http://kaukoluwiki.opendfki.de`

The resources and properties are matched against RDFS-label values defined in the ontology and the user's PIMO. The exact syntax differs from implementation to implementation, after my first implementation in 2003, lots of alternative implementations were published [20] .

6.4.5. Drop-Box for Filing

As identified by Indratmo and Vassileva[IV05] and earlier Barreau and Nardi[BN95], filing information is a crucial task in personal information management. Receiving many documents can cause problems for a user as he often cannot immediately decide about the usefulness of a certain document.

Experience shows that these documents are filed somewhere in the filesystem, with the purpose to categorize them later, but in most of the time this does not happen and the document is nearly lost. This raises the need for an application that relieves the user from the burden of making these decision.

These requirements lead to a use case for the gnowsis Semantic Desktop, that defines the need to (semi-) automatically move and classify a file. This is implemented in a prototypical GUI called *DropBox*[21]. It consists of a folder that is observed by the gnowsis system. If a file is dropped in, a window appears to classify the file (as shown in figure 6.11). The DropBox makes

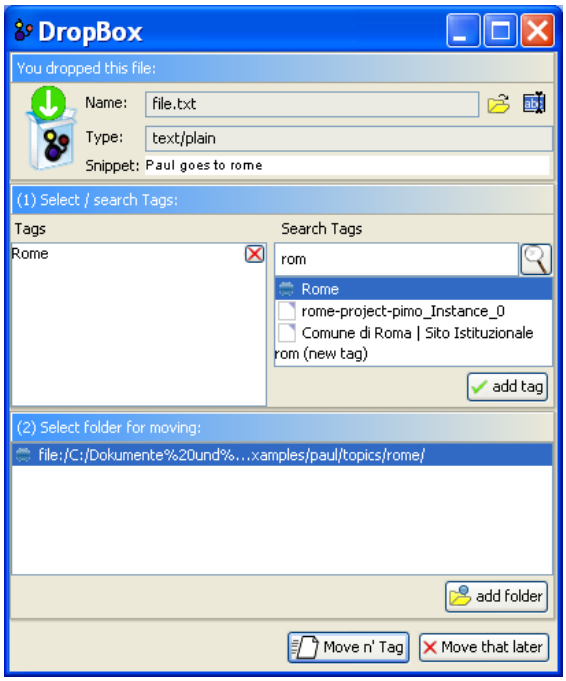


Figure 6.11.: The Gnowsis Drop Box

[20] Most of them can be found in the workshop proceedings linked from the community page about Semantic Wikis at `http://www.semwiki.org/`.

[21] The name is derived from the Mac OS drop-box.

use of the users personal ontology to find out where the document could be moved to and to what other things it may be related to. If the user dropped a file, the data wrapper services analyses it and tries to find similar documents that have already been stored in the, using the categorization service. Note that the training of documents to concepts has to be done beforehand. As result of this matching the system can suggest which elements of the PIMO match against the *dropped* file.

These suggestions are presented in the GUI as shown in figure 6.11 (left side). Additionally, the user has the possibility to add or remove tags. Having chosen the adequate tag(s), the system checks if one or more of the selected tags have a *hasContainer* relation. If so, these containers are presented as suggestions where to move the file and the user only needs to chose the one he likes most. If not, the user needs to add at least one *hasContainer* link as shown in figure 6.11 (right side). This enables the DropBox to move the dropped file as well as relate it to his personal ontology, as soon as the user confirms the settings. Multiple tags can be added to the file. By using the drop-box frequently, the categorization service is further trained.

The software posture of the DropBox, as already described above, is a transient one. This means the application appears only when needed and is in use only for a short time. After the work is done it disappears immediately letting the user continue his work.

6.4.6. Tagging Plugins

One use case for gnowsis was to bring tagging to the users local desktop and provide possibilities for tagging websites, e-mail and file. A prototypical GUI was developed (figure 6.12) that serves as a plugin in Mozilla's e-mail client Thunderbird and provides possibilities to relate incoming emails with tags (things) from the user's personal ontology (PIMO) or with new tags that will be created and automatically integrated to the ontology. The user interface depends on the tagging service to get all existing tags (things) from the users PIMO. As the users starts to type a Letter, the plugin automatically lists adequate (existing) tags from the users personal ontology. With the API of the tagging service, the new assigned tags are then stored.

The software posture of the Tagging Plugin, as already described above, is a transient one. This means the application appears only when needed and is in use only for a short time. After the work is done it disappears immediately letting the user continue his work.

6.4.7. Semantic Search

The gnowsis desktop search (Figure 6.14) is a browser based search visualization that comes along with gnowsis. It allows a desktop search over the users PIMO and all indexed semantic data. Its main input is a fulltext-search field as known from common desktop search tools (e.g. google desktop). The search reaches across all parts of the PIMO ontologies, domain ontologies, and crawled resources. Internally, the search engine visualizes the search results as returned by the *search service* (see Section 6.2.4).

Whereas the quick search (figure 6.13) only displays all search results in a simple list, the browser based visualization categorizes the results in various classes (e.g. persons, concepts) and

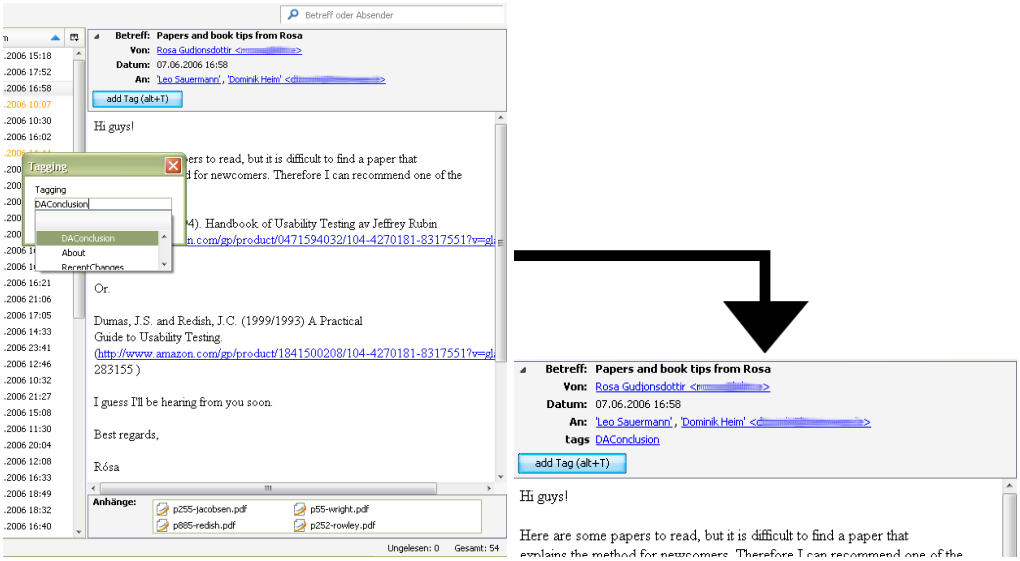


Figure 6.12.: The Gnowsis Tagging Plugin

presents a list of detailed search results to the user. Additionally the most important categories are displayed on top so that the user can get a quick overview over the search results without having to scroll the whole results. The visualization of the search results shows the individual hits clustered by type of the hit. Four concepts (Person, Project, Concept, Event) were selected from the PIMO-Mid ontology as important clusters. The screenshot shows an inferred result included by rules (Heiko Maus is member of EPOS), together with an explanation (the grey text "manager of project"). As displayed in 6.14, hovering over a result displays all its semantic relations on the right side of the according element (in yellow). The ability to show related information based on semantic annotations is new to a desktop search engine.

6.4.8. More Applications

Additionally to the presented applications, there have been many more applications built in co-operations with other researchers, who used the gnowsis platform as a basis for their research.

Contag ConTag[ASRB07, Hor06] is an approach developed by Benjamin Adrian in his Diploma Thesis to generate semantic tag recommendations for documents based on PIMO ontologies and public Web 2.0 services. He designed and implemented a process to normalize documents to RDF format, extract document topics using Web 2.0 services and finally match extracted topics to a PIMO of a user. Due to ConTag we are able to show that the information provided by Web 2.0 services in combination with a Semantic Web ontology enables the generation of relevant semantic tag recommendations for documents. The main contribution of this work is the choreography of Web 2.0 services and an intuitive user interface for document annotation.

SQAPS Norberto Fernandez started with the assumptions of the PIMO which models the main concepts involved in the daily activities of a person: places, organizations, persons, etc. But in

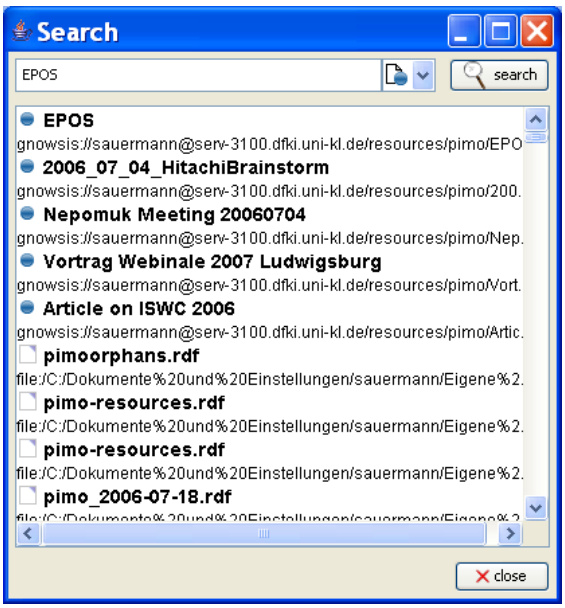


Figure 6.13.: Gnowsis Quick Search

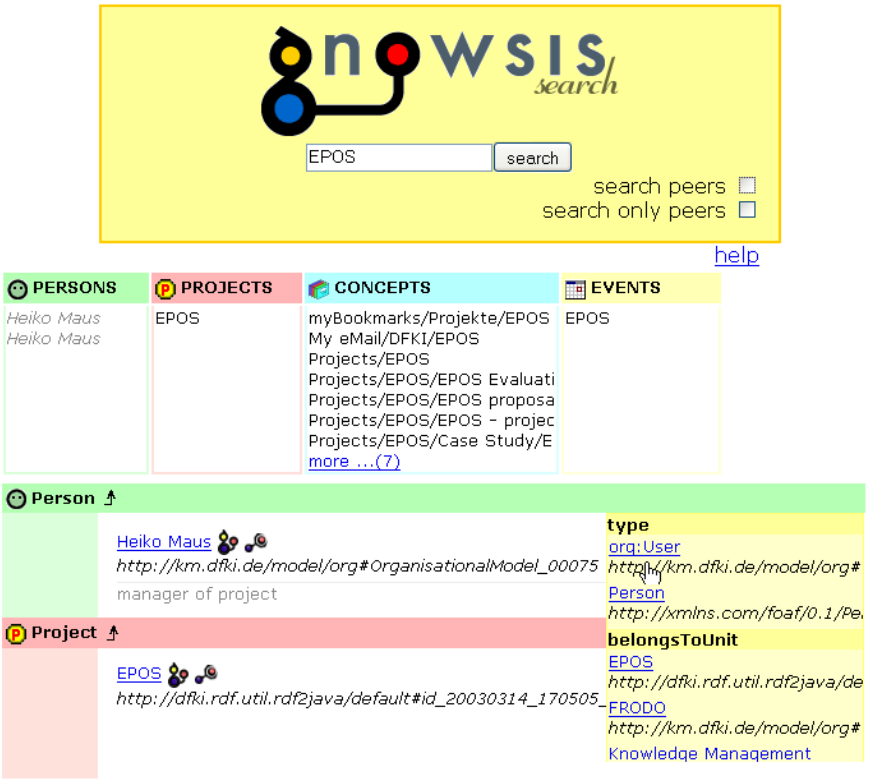


Figure 6.14.: Browser Based Search of Gnowsis

order to be fully useful for a certain user, it needs to be personalized and populated, adding more classes and concrete instances of the existent classes. As the process of manual population could be tedious and time consuming, he proposed an alternative coined *SQA4Desktop* which tries to exploit the information that the user provides while performing Web searches [FGSSB06]. Basically the system requires from the user the annotation of his/her query by associating a concept or set of concepts to it, providing a computer friendly description. The concepts involved in this process need to be taken from an ontology or other semantic source, which is in this case the Wikipedia. So the system invites the user to use Wikipedia articles to express his queries. The concepts from Wikipedia are again represented as things in the user's PIMO, and can be used to annotate resources found in the web search. Apart from populating the PIMO, the approach is useful in resource annotation.

SeMouse

If we look at existing software to edit a PIMO, we can interpret current filesystems and databases as predecessors of the integrated PIMO, making every desktop application a potential PIMO editor. This approach was followed by Jon Iturrioz, Sergio F. Anzuola, and Oscar Díaz in their work on *SeMouse* [IAD06]. Their work reports on the experiences made by turning the mouse into a semantic device, which is used to connect text editing software with an ontology. In existing text editors, like web-browsers or Microsoft Word, the edited documents can be annotated (metadata is exported to the ontology) or authored (metadata is imported from the ontology and inserted to the document). In figure 6.15 the type of a document is annotated, this menu was invoked with the middle mouse button. In the next figure 6.16, the user added the title to this document. Note that the application used, *FoxitReader*, was not manipulated, it works on an operating system-level for many applications.

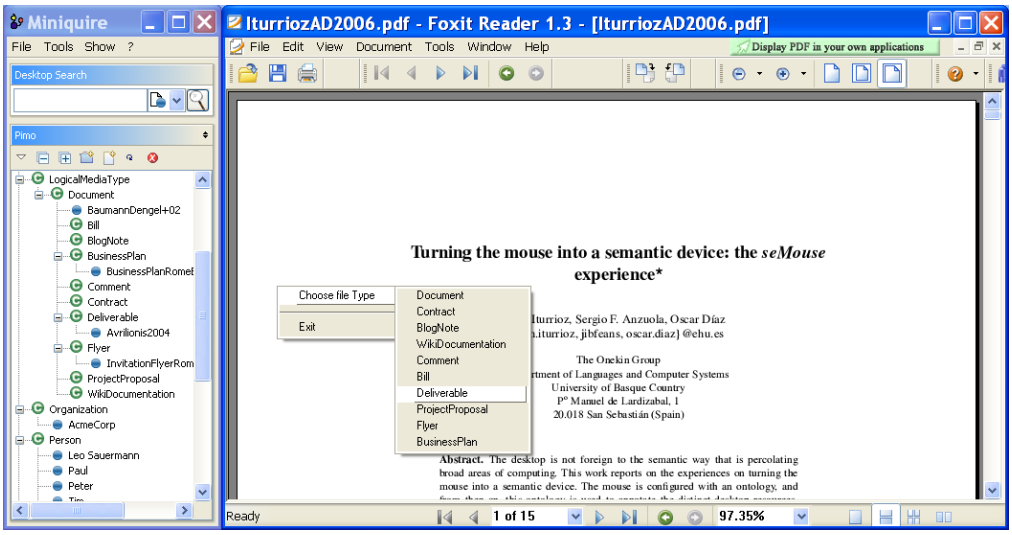


Figure 6.15.: Annotating the type of a document with SeMouse

Because the system is editor-independent, it can handle a myriad of desktop applications that are used in daily information work. *SeMouse* is an approach that augments existing applications using an operating system plugin. In the shown screenshots, *SeMouse* interacts with the gnowsis

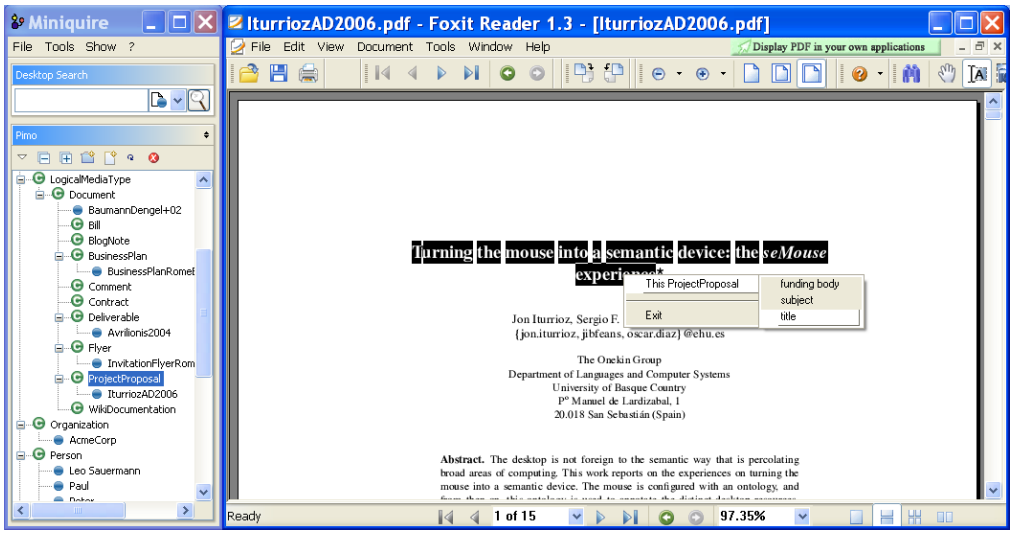


Figure 6.16.: Adding the title of a document

prototype to manipulate the user's PIMO. This approach shows us that we can embed annotation functionality to any text editor with affordable investments.

Wild Experiments

Additionally to the applications stated beyond, which were evaluated and tested in practice, we did create many more applications and mash-ups of Semantic Desktop technology with other applications. The motivation behind these applications was mostly, to *mash-up different applications to see what result comes out at the end*, a typical web2.0 set of mind. As an illustration how PIMO and the services should be used, a short hint on the features and possibilities is given:

- A Microsoft Outlook plugin that shows the related information to address book entries.
- A modified online roleplaying game[22] was modified to allows players to interact with their PIMO as virtual objects, for example rendering friends as avatars.
- Various mash-ups were programmed with other developers, combining FoafNaut with gnowsis or fenfire with gnowsis, offering graph-based user interfaces to the PIMO data[23].
- A Microsoft Powerpoint plugin was created, allowing to link elements in Powerpoint presentations with PIMO things, clicking on the element then opened the PIMO thing.

The scientific relevance of these projects is minor, but the practical implication is: for all of these projects, it was possible to integrate two software system, which before were not thought to be integrated, within one day and achieve a useful result. The cost of integration of personal information management applications can be significantly lower using Semantic Web technologies, which is important for the application of our architecture at large.

[22] In an experiment, an open source clone of the Ultima Online server was modified. `http://leobard.twoday.net/stories/4528125/`

[23] Foafnaut: `http://leobard.twoday.net/stories/307895/`
Fenfire: `http://leobard.twoday.net/stories/406584/`

6.5. Summary on Architecture

Based on the semantic services underneath, the applications provide various opportunities for cross-application PIM. The data from various applications was integrated with the PIMO allowing the user to keep track of the same idea independent of task.

In the gnowsis prototype we have created several innovations in the field:

- The first personal semantic wiki (and perhaps the first semantic wiki at all, published in 2003 [Sau03]).
- A Semantic Desktop architecture providing services instead of only applications.
- A sound concept to extend existing applications with plugins, using tagging as a metaphor.

The listed applications are examples and we presented only the features evaluated in user studies. Many more application scenarios have been realized at DFKI and by fellow researchers.

By providing the services and applications, we have shown that known problems and needs of Personal Information Management can be addressed using the integration provided by the PIMO and the Semantic Desktop. Before PIMO, the data structures for integration existed, but had no clear instructions how to integrated typical PIM data (e-mails, documents, appointments). The identified services are crucial elements for PIM, they address the core requirements of PIM, independent of application domain or application. The shown applications have been evaluated, as shown in the following chapters.

In principle, related PIM research approaches can be realized now to the Semantic Desktop, for example the innovative user interface of *Haystack* [QHK03], the time-based metaphor of *Lifestreams* [FG96], or the all-inclusive approach of *MyLifeBits* [GBL+02]. Other researchers already started to create PIM applications based on Semantic Desktop services and ontologies created by us, more may follow motivated by the detailed results created within the NEPOMUK project.

CHAPTER 7
The Gnowsis Semantic Desktop Prototype

In this chapter, the *gnowsis reference implementation of the proposed model and architecture* is presented. *Not all services are covered in detail*, a focus is given on services that were complex to realize.

The web has always had a benefit from open source reference implementations of theories. Tim Berners-Lee used a reference implementation of the first web server and web browser to advertise his ideas. Reference implementations are an instrument of the World Wide Web Consortium to test ideas [1]. They help to design a new idea, understand it, and document it in parallel with a recommendation document.

The gnowsis prototype was used in different ways.

- First, it validates that the architecture proposed in Chapter 6 and the underlying model proposed in Chapter 5 are implementable and do work as intended. This validation is purely empirical. In Section 7.4 a summary of this validation is given.
- Second, the prototype is used in a formal validation including end users. These experiments with the prototype, and their results are described in the Chapters 8-11. The experiments validate the scientific question of this thesis.
- Third, the prototype was used personally by myself and the other scientists working on the Semantic Desktop idea to support our own personal information management and to learn about the software by "eating our own dogfood".
- Fourth, other researchers besides us had free access to the open source software. This allowed them to verify our findings and to extend our work with their own ideas (see Section 6.4.8 for work by others that used our software).

The first gnowsis prototype was started in March 2003 as an implementation for my diploma thesis [Sau03], and then continued as a project at DFKI. The findings and results have been published in several papers, and four diploma theses (five, including mine). **The source code and distributions of gnowsis can be downloaded on the project website** [2].

In this chapter, we will describe the prototype version 0.9.*, which is a beta version created in 2006. The evaluation of the personal semantic wiki was done with this implementation. The older version, 0.8.3, was used for the evaluation at Siemens Business Services, components of it

[1] `http://www.w3.org/2005/10/Process-20051014/process.html#cfi`

[2] Website: `http://www.gnowsis.org`
For developers: `http://gnowsis.opendfki.de`

will only described when they are not present in the later version [3]. Focus will be on the *non-visible background services* such as the RDF store or the Aperture framework for data wrapping. The visual interfaces are self-explanatory and documented in videos available on the web, also the quality of user interfaces will improve during the NEPOMUK project.

7.1. Used Software

To realize our work, we build upon the shoulders of giants. Software libraries "galore" were used to realize the gnowsis Semantic Desktop. First we give you pointers for the most important parts, which were the core of our project. As part of our implementation, we also started new open source projects such as *Aperture* and contributed considerably to existing ones.

Java is the programming language used to develop gnowsis. In version 0.8, we restricted ourselves to Java 1.4. For 0.9, elements of Java 1.5 were used.

Kaukoluwiki is a semantic wiki developed by Malte Kiesel from DFKI. It enhances the JSP based *JSP wiki* with semantic web features. Kaukoluwiki was extended to be compatible with the gnowsis platform.

Jena is a semantic web API developed by HP Labs. It features an RDF storage layer, inference support, querying using SPARQL, an RDF parser, and other features.

Sesame is a semantic web API comparable to Jena. We used the 2.0 alpha version, which misses many features but provides support for context (quads) from the ground up. In practice, sesame was used as the RDF database, and Jena as a frontend API to interact with it.

Lucene is a fulltext indexing and search engine available in Java. It was used to provide fulltext search functionality.

LuceneSail is a project initiated by me and Christian Fluit to provide fulltext indexing capability to RDF stores.

Aperture is an open source Java framework for extracting and querying full-text content and metadata from various information systems (e.g. file systems, web sites, mail boxes) and the file formats (e.g. documents, images) occurring in these systems. The project was initiated by me and Chris Fluit to share the effort between multiple companies.

[3] Version 0.9 is a complete rewrite, not all functionality of 0.8.3 was ported to it.

Apache XML-RPC is an implementation of "XML-RPC", a http-based remote procedure call protocol that uses XML to encode messages. It is a much simpler and lightweight approach compared with SOAP.

Brainfiler is a commercial text indexing, search, and classification engine developed and marketed by Brainbot AG. It was used to classify documents based on text similarity and provides group access to documents.

EPOSWorkspace is a service-oriented architecture to assemble applications based on multiple services. It was implemented by Andreas Lauer of DFKI, and later extended by us both together to support web applications.

GnoGno Components is a framework of GUI components that allow creating RDF applications based on Swing using the Eclipse Visual Editor. It supports drag-drop, lists, trees, and editor components. This project was initiated by myself.

To give an impression what libraries are used underneath, here is the list of software packages that have been integrated by us into the services and applications. Without these freely available open source libraries and tools, realizing our work would have been impossible, and we want to show our respect to the authors by listing the projects here.

- Ant for packaging
- Apache Commons: Codec, Fileupload, httpclient, io, logging
- Concurrent for multithreading
- Demork for parsing Thunderbird Address Books.
- FlickrAPI to connect to flickr.com
- Glazedlists for the user interface
- Htmlparser to read HTML files
- Jacob to interact with Microsoft Outlook
- Java Actication Framework.
- Java Mail API to connect to IMAP mail servers
- Java Servlets, Server Pages and the JSTL standard tag library by Apache
- Java Swing for the user interface
- JGoodies-looks for the user interface
- JNIWrapper for system tray

- Json for web-serialization
- JUnit for Unit testing
- PDFBox to read PDF files
- POI to read office documents
- Tablelayout for the user interface
- Wizard-framework for the installation menu

The application was developed using the free *Eclipse* Rapid Application Development (RAD) environment , mostly using version 3.2. The *Eclipse Visual Editor* proved very useful to create the user interface in a point-and-click way.

Licensing

License of Gnowsis The gnowsis implementation is published under the BSD license, which is a liberal free software license.

For Aperture, the core project APIs and architecture are licensed under the Academic Free License (AFL) version 3.0[4]. It allows developers to license implementations of these APIs under any license they see fit including proprietary and commercial ones.

The implementations of these APIs contained in Aperture distribution are licensed under the Open Software License (OSL) version 3.0[5], which is a reciprocal license. This effectively means that changes to these implementations have to be made available to the community, while these implementations and their derivatives can be used in applications licensed under a different license.

7.2. Service and User Interface Implementations

In the following sections, the individual implementations of the services are presented. Each service from Section 6.2 was realized, but only those with considerably implementation effort are presented now. For example, the **PIMO Service and Tagging Service implementations are not listed** because they were a rather straightforward implementation of a set of methods. Also, focus is given on the engineering optimizations and the challenges during implementation, to allow others to replicate our results.

At the end of each section, the results are compared to the proposed architecture. This shows how the implementation empirically verified the proposed architecture. Some services were innovative in their implementation and started long-lasting software projects, where this happened we give a short summary on the outcome to illustrate the quality of the architecture. Again, the reason why the PIMO and Tagging Services are not discussed is simple: The set of methods

[4] http://www.rosenlaw.com/AFL3.0.htm
[5] http://www.rosenlaw.com/OSL3.0.htm

defined in these services proved to be a minimalistic ideal, they were easy to implement and exactly sufficed the needs of the user interfaces – a perfect match.

For all implemented services, the sourcecode and documentation is available online [6].

7.2.1. Personal RDF Store Implementation

The central component of the prototype is naturally an RDF storage repository. Some requirements have to be met:

- The store has to provide **server interfaces** for reading and writing information from different applications on the desktop.
- For importing and updating external ontologies, it has to be possible to add and remove ontologies as sub-graphs in the store. This can only be realized efficiently using a **quad store**.
- Optimization of storage aims at **fast reading, slow writing**. User interfaces have to react fast on displaying information, changes can take time.
- The store has to support **fulltext search** that is capable of basic information retrieval methods like ranking the results and similarity search.
- An inference engine should be used to create additional triples based on the knowledge in the store. These closure rules can provide **inference** on the ontologies (sub-class, sub-property relations).

The requirements were met by the local desktop RDF store implementation. Gnowsis uses four different databases inside the store.

1. The **PIMO store** handles the information in the user's *Personal Information Model* (See Section 5.2.2).
2. The **resource store** handles the data crawled from Aperture data-sources (See Section 7.2.4).
3. The **configuration store** handles the data about available data-sources, log-levels, crawl-intervals, etc.
4. Finally, the **service store** handles data created by various gnowsis modules, such as user profiling data or metadata for the crawling of data-sources.

The **PIMO store** stores the user's Personal Information Model. As described in section 5.4, this includes the imported upper and mid-level ontologies, domain ontologies, one personal information model of the user (user-PIMO), and additional ontologies can be imported or removed.

[6] Source code: `http://gnowsis.opendfki.de/repos/gnowsis/tags/0.9.3/`
Documentation: `http://www.gnowsis.org/statisch/0.9/doc/javadoc.html`

User generated annotations about native resources (such as tags or relations) are stored in the PIMO-Storage. It supports fulltext-search so that users can quickly find concepts they need; inference to entail basic subclass, subproperty, and inverse-property rules; and named graphs to make the handling of multiple ontologies efficient.

Data and metadata about native resources is stored in the **resource store**. It is dedicated to information coming from **external sources**. Files, e-mails, RSS feeds, or other structured information sources are examples. This store is designed to handle a larger amount of data, and allows fast storage and deletion of data. The user cannot change data inside this store (in terms of changing single triples or adding annotations), it is dedicated to serve as search store and a basis for further deductions that can help to fill the PIMO storage.

Separating the PIMO store from the resource store was an important decision for the gnowsis architecture, and it was made for several reasons: The resource store is inherently chaotic, since it mirrors the structure of the user's applications (consider your email inbox), whereas the thoughts (eg, concepts and relations) can be structured separately in the PIMO. Another reason was efficiency, while a user's PIMO may contain a few thousand instances for a very frequent user, it is not uncommon for people to have an archive of 10,000 emails or 50.000 files. By separating the two we can save time and resources by only performing inference on the PIMO store. We also note that a similar approach was taken in many other projects, for instance the topic maps community, where topics and occurrences are separated [Rat03].

The storage modules in gnowsis 0.9 are based on Sesame 2[7] and are using Sesame's native Storage And Inference Layer (SAIL) to store the data on disk. In the previous gnowsis versions we used MySQL in combination with Jena as triple store, but this enforced users to install the MySQL database server on their desktops and also the performance of fulltext-searching in MySQL/Jena was not satisfying. By using Sesame2 with the embedded native SAIL we simplified the installation significantly. In addition to the raw RDF the PIMO and resource stores use an additional SAIL layer which utilizes Lucene[8] to index the text of RDF literals, providing extremely fast full-text search capabilities on our RDF stores. Lucene indexing operates on the level of documents.

LuceneSail text index

Our **LuceneSail** has two modes for mapping RDF to logical documents: one is used with Aperture and will index each Aperture data-object (for example files, webpages, emails, etc.) as a document. The other mode does not require Aperture. Instead one Lucene document is created for each named RDF resource in the store. The resulting Lucene Index can be accessed either explicitly through Java-code, or by using special predicates when querying the RDF store. Figure 7.1 shows an example SPARQL query for PIMO documents containing the word "rome". The LuceneSail will rewrite this query to access the Lucene index and remove the special predicates from the triple patterns. This method for full-text querying of RDF stores is equivalent to the method used in Aduna MetaData server and Aduna AutoFocus.

The **inference on the PIMO store** was realized using an *adapted rule-based forward chaining inferencer*. We have based our implementation on the Jena inference engine, which provides a generic rule inferencer. The way Jena works was not fully compatible for our approach, as we

[7] See the project homepage at `http://www.openrdf.org/`.
[8] `http://lucene.apache.org/`

```
SELECT ?x WHERE {
  ?x rdf:type pimo:Document ;
    lucene:matches ?q.
  ?q lucene:query ``rome''.
}
```

Figure 7.1.: A Query using special predicates for full-text searching

had the case of frequent deletion and modification of data. Jena, as many other inference engines, discards all inferred knowledge and runs the inference again when any triple changes. This may be acceptable for some cases, our underlying fulltext index and the amount of data and our requirement for fast answer times when the user interacts forbid this. We adapted the rule engine to work on top of the Sesame2 store and changed it to support **transaction-based inference**. One transaction consists of multiple read/write operations adding and removing statements. The Jena inference engine is triggered when triples are inserted. We adapted our store to first add triples and remember which triples are inferred by the rule engine. The inferred triples are then stored in the database. When deleting facts, the same rule-based engine is used to infer which triples would have been added assuming the triples were added, these were then removed. This worked amazingly well for the evaluation and our implementation in general.

In general, we used Jena's notation of **forward chaining rules** (which are similar to Horn clauses). Additionally, we **limited inference to the TBox** and stored the **entailments in the database**. This is a simple approach to optimize RDFS inferencing, only the rules for sub-classes, sub-properties and inverse properties were actually inferred and stored. On query time, applications had to expand the queries to include inferred sub-classes or not.

For example, to get all instances x of the class y, the SPARQL query shown in Figure 7.2 was used. This adds one more tuple to the queries (the sub-class tuple) which is an acceptable effort. Also, the queries are compatible with RDF inference engines that realize full inference on ABox and TBox.

```
SELECT ?x ?y WHERE {
?x rdf:type ?t.
?t rdfs:subClassOf ?y.
}
```

Figure 7.2.: A query using limited TBox inferencing

7.2.2. Use of Named Graphs in the Store

Named graphs can be used to give a statement in a RDF store a fourth identifier - the context. As each statement that usually has three items (subject, predicate, and object) now has a fourth item, the context identifier, these statements with four-items are also called **quads**, whereas normal statements are usually called **triples**. A storage server that is capable of handling the fourth item is then called a **quad store**. If many triples in the store get the same identifier, one can assume

that through this they belong to the same graph, while triples with other identifiers belong to another graph in this store, hence the term **named graphs** is used to state that one database can store more than one graph in a database, and the graphs are kept separated by the identifiers. Detailed information about named graphs was published by Carroll et al in [CBHS05]. Given this capability, they identified that context information in a named graph can carry different meanings.

- Provenance: The context identifies the filename/url where the triple was downloaded from. The URI stored in the context is a URI identifying the provenance of this triple. For each graph in the quad store is one context, used to keep track of provenance information and provenance chains.
- Restricting information usage: Information providers might want to attach information about intellectual property rights or privacy information, to restrict usage of the published information. The context would then refer to a privacy rule.
- Access control: Like file permissions, users may want to allow fine grained access control. The context identifies an access rule, similar to privacy rules.
- Signing RDF graphs: To build a web of trust, signatures are the basis to verify the author or other information about a graph. Signing the graph is based on provenance.
- Stating propositional attitudes such as modalities and beliefs
- Scoping assertions and logic where logical relationships between graphs or triples have to be captured
- Ontology versioning and evolution: based on provenance information, an ontology can be replaced with a newer version or changes synchronized.
- Identifying individual triples: if a combination of above methods is needed, each triple can get an individual context and then the provenance or access control is expressed on each triple, similar to reification. The context contains a unique identifier, preferably a perfect hash [CLRS01] to assume that identical triples get the same identifier.

The problem of named graphs is, that the context cannot be used for more than one purpose. Using the context for provenance restricts usage for access control. By giving each triple a unique identifier, both can be achieved, on the cost of a decreased performance and increased storage space. As many uses for named graphs reference to provenance, we decided that **named graphs are used to store provenance information in the PIMO storage**. Each ontology and domain ontology is a named graph in the store. This allows effective ontology versioning and management of ontologies by adding new domain ontologies. Also, import directions can now be evaluated, if an ontology imports another ontology, the inference engine can explicitly evaluate this statement. This allows to check ontologies for correctness: if an ontology uses elements of other ontologies that are not explicitly imported, the PIMO checker will detect that.

Discussion of the RDF Store Implementation The presented implementation covers all features of the RDF Store service as defined in Section 6.2.1. Additionally, it implements inference in a performant way and also supports fast inference during the deletion of triples, which distinguishes our implementation. The store implementation shows that, using free software and good engineering, the presented service can be implemented within a short time effort realizing enough quality to support real users. Implementations with commercial-quality can be based on these implementation decisions.

7.2.3. URI Identifiers for the PIMO

For the gnowsis implementation, a URI scheme and naming convention had to be created to represent personal resource identifiers for things in the PIMO.

HTTP is not a good choice in this regard, as it requires to have a correct DNS hostname as part of the URI, whereas most desktop users neither own a domain name nor can the IP addresses of their computers be registered at DNS. The common desktop users are working behind firewalls in home networks or at companies, which would also not allow to host webservers on their desktops. Neither are the IP addresses of computers accessible that access the internet behind a NAT router, which is the case for most users. To sum it up, if we had taken a HTTP scheme, it would have looked somehow like this:

```
http://paul.example.com/user/paul/thing/Rome
```

Given a Semantic Desktop running on Paul's desktop computer, this URI would probably not be accessible nor feasible because:

- Paul or his company would have to buy a domain (example.com).
- Paul would need a sub-domain name for his computer (paul.example.com).
- The IP address of his desktop computer would have to be registered. Given the fact that most computers today are laptops, which frequently change IP address, this is a problem.
- The IP address of Paul would probably be that of a private network (such as 192.168.0.10), which overlaps with others.
- Given we want to create a Social Semantic Desktop with distributed services, other users would not be able to access and dereference the URIs because of NAT or firewall configurations.

Therefore we decided to use a different protocol, the peer-to-peer protocol XMPP (jabber). This allows any user to host the RDF resources where ever they want. If they are online, via the jabber protocol, if they are offline via the jabber server.

The URI scheme functions as follows:

```
gnowsis://[jabber id]/resources/pimo/

example: gnowsis://paul@example.com/resources/pimo/
```

The different parts are:

- gnowsis - a Semantic Desktop implementation
- jabber id - the jabber-id of the user (typically username@jabberhost)
- resources - the place for rdf resources
- pimo - the pimo of the user

Discussion of the gnowsis URI-Scheme The gnowsis URI-scheme provides a globally unique (the user is part of the URI) but locally adaptable (Semantic Desktop software can generate new URIs) URI scheme to identify PIMO Things. It was crucial to develop this scheme to realize the identification methodology described in Section 5.4.6. Having a new scheme identifier ("gnowsis") is crucial to register the URI scheme with the operating system and bind it to the Semantic Desktop daemon services[9].

This URI scheme was straightforward for our case and may be usable for other applications as well. Even if the user's do not employ the possibility of P2P communication, the idea of the namespaces are practical. For the Semantic Desktop vision and other reasons Malte Kiesel, Frank Osterfeld, and Sven Schwarz created the advanced Jabber server *nabu* [OKS05] which is extended with RDF features.

7.2.4. Data Wrapper: The Aperture Framework

Harness Existing Data

To interface with applications that are not semantically enabled gnowsis uses a framework called Aperture [Ape05]. Harvesting as much existing semantic information as possible from legacy applications benefits the user as it lowers the entry barrier to use semantic applications, e.g., when compared to approaches that rely on manual annotations.

Obtaining this information is a complex engineering task. The information is spread among a variety of source types and file formats. Crawling and indexing this information does give us these capabilities at the cost of having to keep the extracted metadata in sync with the original sources. Nevertheless, the operating system providers implement the same approach with Windows Desktop Search or Apple Spotlight. These approaches are limited to fulltext indexing and simple key-value metadata and are operating system-specific.

Core functionality

The Aperture project provides various open source components that harvest and index fulltext and metadata of native resources. The core functionality provided by Aperture is crawling data sources. It is described also in Section 6.2.2 and a schematic overview is given in Figure 6.3 above. The crawling process involves iterating over all data items available in a data source, extracting the data, converting it to RDF triples and forwarding it to the client application for further processing.

[9] Registering the scheme with the operating system URI handling service was not done in the prototypical gnowsis implementation, but commercial implementations should do it.

The use of RDF may have its price as programmers may find it relatively hard to handle RDF models, compared to simple Java Maps with key-value pairs. To alleviate this problem we have designed a simple Java interface called RDFContainer for handling small RDF models. RDFContainer instances are typically used to move metadata from the extractors to other components. In gnowsis, the extracted data is stored directly (without further inference) in the resource store RDF database. A separate thread is started to crawl the datasources and store the extracted data in the RDF Store, (into the **resource store**).

Connecting Aperture to gnowsis

Available crawlers include:

File system crawler Crawls file systems. It extracts basic metadata about files (names, sizes, dates of the last modification, etc.). It also reflects the structure of the directory tree. The information returned by the crawler can be augmented by other components of the framework, as we'll see further.

Web crawler Crawls web sites. It extracts basic metadata about files available on the web, by using the LinkExtractor service to extract and follow links from an HTML document. The information returned by the web crawler can also be augmented by other components of the framework, as shown next.

IMAP mailbox crawler Connects to an IMAP mail server and crawls the mailbox in search of newly arrived e-mails. It extracts the metadata fields such as *From*, *To*, *Cc*, *Bcc*, etc. It can be used to make a semantic application aware of incoming mail.

Calendar crawlers Crawl calendaring data, as newly added events, todos and journal entries. For events, it crawls iCal calendar files. The iCal format is widely accepted, and is specified in the RFC 2445 document[10]. Many calendaring applications either use it natively or provide functionality to export their data to this format. For todos and journal entries, it crawls personal calendars stored by Microsoft Outlook.

Address book crawlers Crawl contacts stored in address books. Currently address books generated by Apple iCal, Mozilla Thunderbird and Microsoft Outlook are supported. They extract individual contacts and annotate them with all metadata that is available (names, addresses, e-mails, phone numbers, etc.). They complement the calendars crawler in providing functionality for complete integration of personal information management applications into semantic environment.

Additional functionality

The Aperture Framework divides the crawling process between multiple components. The crawler itself usually implements only the crawling logic. The handling of individual files is provided by plugins:

DataAccessor The URI of a resource is fed into a scheme-specific DataAccessor which retrieves the resource and creates a data structure for it, holding the binary data, as well as any metadata provided by the scheme e.g., file names and last modification dates.

[10] `http://www.rosenlaw.com/AFL3.0.htm`

MIMETypeIdentifier Uses the binary data (which could be obtained by a DataAccessor) to determine the MIME type of the contents.

Extractor Extracts the full text and detailed metadata from a binary file. It is implemented for a specific file format, so usually an identification of the MIME type is necessary prior to application of an extractor. Extractors for following file formats are provided:

- Microsoft Office documents (Word, Excel, PowerPoint, Visio)
- Microsoft Works
- Microsoft Publisher
- Corel WordPerfect Office (WordPerfect, Quattro Pro, Presentations)
- OpenOffice
- Outlook, Thunderbird and Apple address books
- PDF
- HTML
- XML

Other functionality Other services provided by Aperture include link extraction, security management and opening documents in their native applications.

Lessons learned

Performance Optimization Before starting the Aperture project, experiments were conducted to learn possible ways to integrate desktop applications. In a series of experiments done during the NEPOMUK preparation phase (in 2005, before the project started) we evaluated whether to use live adapters that do not crawl data from data sources and buffer the data inside an index, or instead contact the data source on demand, when a query is asked to the system. This latter approach was called a *virtual RDF graph*. A virtual RDF graph is an adapter software that connects to the data source (such as Microsoft Outlook or the file system) and only extracts the data when needed. To do this, a query is passed to the adapter which analyses the query and determines how to extract the needed information and then reads the information from the source, transforming it to RDF on the fly. [SS05a] describes the experiments made for comparing virtual RDF graphs to crawling frameworks. From the performance side, query time was comparable between both approaches for simple queries. Virtual RDF graphs seemed to be a promising approach, especially for data sources that allow a generic access to the data and include search functions (like SQL servers or SPARQL-enabled data sources. In general, we found these drawbacks:

- The effort to implement virtual RDF graph adapters is higher than crawlers because of their dynamic nature.
- Only simple queries gave satisfying answer times, more complicated queries (like nested SPARQL queries) were both hard to implement and if implemented, response times varied from milliseconds to hours.

For these reasons, the decision was taken to implement Aperture as a crawling framework.

Discussion of the Data Wrapper implementation Aperture fulfills all requirements of the designed Data Wrapper service as described in Section 6.2.2. Based on our published experiments [SS05a], excellent engineering, and cooperation with industrial partners, it was possible to establish the basis for a commercial-quality framework.

The Project

The Aperture project was initiated as independent library as a project at sourceforge[11]. It is a result of close co-operation I initiated between the German Centre for Artificial Intelligence (DFKI) and Aduna—a software company headquartered in Amersfoort, Netherlands. The project began in 2005, when I recognized a common need of both organizations for the same Java Framework for wrapping non-semantically enabled applications and data formats.

The implementation is described briefly in [SGK+06] and in more detail on the Aperture homepage [Ape05]. From 2005 to 2008, Aperture has been downloaded more than 10.000 times, was used in many other projects, and was adapted into the industry-grade information extraction suite Eclipse SMILA [12]. The core architecture is the same as described in this thesis. The uptake of the architecture into industrial applications is a good validation of it's quality.

7.2.5. Categorization Service

There are two alternative implementations for the categorization service. The first and more powerful in terms of functionality and performance is based on the product *Brainfiler* by *Brainbot AG* Mainz [AG] (this was used in gnowsis 0.8.3 and for the evaluation in Section 11). Brainbot is a DFKI spinn-off company, the categorization algorithm is based on a TF/IDF matrix of all crawled documents and similarity measures working on this matrix. The system supports creating categories of documents, a category is trained by adding an initial set of documents and then the system can suggest which category fits for new documents. The interna of the categorization scheme are not published.

The second implementation is less powerful, and based on the open-source *Lucene* text indexing engine (this was used in gnowsis 0.9.* and for the evaluation in Section 8). It keeps an internal TF/IDF matrix of all extracted documents, similar to Brainfiler. Categorization for a new document works based on a similarity search with the new document as input. The similarity search returns documents from the store in a ranked order, highly similar documents are returned first. Each returned document is checked for annotations, these existing annotations are suggested for the new document. The algorithm is primitive but suited good enough for the needed functionality used by the *Drop-Box*.

Discussion of the Categorization Service There are different ways how to implement the categorization service and we were able to try both a commercial implementation and to provide our own open-source implementation based on Lucene. Our implementation covers

[11] `http://aperture.sourceforge.net`
[12] `http://www.eclipse.org/smila`

all core aspects of the categorization service as designed in Section 6.2.5. As it is fully based on free software, an evaluation with more people was possible. Because it is based on commercial-quality software (the Lucene engine), it provides high performance.

7.2.6. User Work Context Implementation

The user work context server was implemented and managed by our colleague Sven Schwarz and is topic of his publications and dissertation [SRB03, Sch05, SS05b, Sch06].

It was also evaluated and documented by Schwarz and other colleagues at DFKI. At this point, we want to stress the fact that representing the current user context and observing the user is a key for context-sensitive applications.

Discussion of the User Work Context Implementation The implementation created for this thesis fulfills the requirements stated in Section 6.2.8. It included many plugins for various applications that monitored user operations and forwarded these to a server application running as part of the Semantic Desktop. The quality reaches a level where the observation plugins do not decrease the user experience.

The implementation is managed by Sven Schwarz and is available for download [13]. We contributed to his effort of creating a long-lasting open source project for user observation and work-context identification.

7.2.7. PIMO Interfaces in Swing and GnoGno

The interfaces Miniquire (Section 6.4.1), PIMO Thing Editor (Section 6.4.2), and Drop-Box (Section 6.4.5) were implemented using Java Swing windows.

Although standards like *fresnel* exist to describe the rendering of RDF information, and this standard has been implemented, there are only limited approaches of creating user interfaces to edit RDF data in a generic way. Today's programming languages and tools give support for user-interface development based on relational databases or XML formats as the underlying data. For RDF, we did not find out-of-the box Rapid Application Development tools with rich widgets to create desktop applications. In fact, we had to create such a widget library ourselves. The Swing widget library was used as a basis to create lists, edit-fields, icons, and drag-drop behaviour. We wrapped these in a framework coined **gnogno gui components**. A summary on the components can be found on the respective website [14], there is also an introduction video showing how to use the components for rapid application developement (RAD) [15].

Discussion of the user interface implementation Using the gnogno gui component framework, we were able to provide a convenience in programming RDF interfaces that

[13] `http://usercontext.opendfki.de`
[14] `http://gnowsis.opendfki.de/wiki/GnognoComp`
[15] `http://www.dfki.uni-kl.de/~sauermann/2007/05/31/edited_web.html`

was before only available to commercial RAD tools such as Delphi. The created user interfaces are modular, tested, fast, and conform to common user interface guidelines. Based on these interfaces, it was possible to conduct the evaluations presented in the next part of this thesis, but also it was possible for other researchers to do their own experiments (as listed in Section 5.8) and to build more software on top (Section 6.4.8).

7.2.8. Personal Semantic Wiki: Kaukolu in Gnowsis

The *Personal Semantic Wiki* is the most promising approach on the Semantic Desktop for knowledge articulation. Knowledge can be expressed either as free-text, as formal annotations using RDFS or in a mixture of both. In above Section 6.4.4 we already introduced the features of the integrated software.

The implementation was an adaption of the existing open-source *JSP wiki*, and *Kaukoluwiki* which builds on it.

They key elements to add have been adaptations in the GUI to call gnowsis functions from within the website (for example to open files on the harddisk or to start the PIMO Thing Editor). Underneath, the storage layer of the wiki was adapted to save the wikipages as RDF statements connected to the PIMO. The semantic wikiparser was adapted to use PIMO things as possible concepts in addition to wiki page-names and RDF resources.

Discussion of the Kaukolu Semantic Wiki implementation The implementation of the pesonal semantic wiki using the Kaukolu project covers all aspects defined in Section 6.2.7. A drawback of this implementation is that it increases the installation size of gnowsis considerably. Also, including a complete application (Kaukolu is intended to be used as a standalone server) into another application was a technical challenge at the beginning and a full merge of both applications was not possible. This caused minor drawbacks for the evaluation (see Section 8.8.1): users described a loss of context when switching from the Swing desktop application to a web-application for editing wiki pages.

Another drawback was the quality of this part of the code. Due to time constraints, some performance issues were never solved in the glue-code connecting Kaukolu to gnowsis. This caused the user interface to get slower as more data was added to the system, a problem that was not experienced in the underlying wiki system nor in gnowsis. It could be fixed by revisiting this code.

7.3. Lessons Learned in the Implementation

Some problems blocked the implementation and the evaluation, where we improved the architecture based on learning our lesson the hard way. In the following section we want to focus on some problems that were apparent during the implementation and that have been partly addressed, or that caused us to change the implementation.

When research on the Semantic Desktop started in 2003, deployment of Semantic Web applications in practical use was generally not known (besides RSS and Mozilla XUL). Although several implementations of the underlying architecture existed, RDF stores, validators, parsers, query engines and inference engines, their quality and features were very limited. For the research prototype, we limited ourselves to open-source implementations, to be able to extend the underlying frameworks when needed[16].

The Semantic Desktop is a challenging domain for RDF, a person's data can quickly sum up to 100.000 triples, crawling a full hard-disk of a long-term user can result in millions of triples. Query answering time is expected to be in the dimension of the 10th of a second, to implement user interfaces that can quickly react to user interaction, as typical on the desktop.

The following drawbacks occurred over the years, we hope that they are overcome by now:

- Triples stores are not enough, for our research we needed to identify the source (provenance) of the triples, therefore quad-stores were a prime requirement. Jena did not implement quad stores for long, Sesame's implementation supporting this is today (Sep 2007) still in the beta phase.
- Fulltext indexing in combination with RDF storage. In user evaluations, we quickly saw the need for full-text indices that are combined with RDF stores. SQL databases support this for longer.
- Immediate (=fast) inference support is needed to really benefit from the semantic features of the semantic web. Available open-source inference engines often implement inference in-memory, assuming that gigabytes of RAM are available on the servers that run the software. Also, answering times are slow[17].

To summarize the drawbacks, it is very hard to find an RDF store that can work with the dimension of data required on the Semantic Desktop and implements inference, and fulltext search, and answers fast. If it also has to be open-source, and able to work on limited desktop resources (RAM, CPU), it is even harder to find one. Until today (September 2007), we were not able to find one.

Based on our experience, we can only recommend to carefully pick the right underlying frameworks before starting a project, and when they do not exist, consider limiting the features and requirements instead of creating your own tools.

The Haystack team were forced to implement all their tools by themselves (back then, no capable RDF tools for Java existed). They implemented RDF store, user interface framework (and widgets), a new programming language, an interpreter for the language, and concrete applications completely on their own. This task is impressive and resulted in an integrated environment.

[16] For example, I implemented a query optimization and reordering algorithm for Jena for the first prototypes in 2003. Using the commercial closed-source RDFGateway, this would not be possible.

[17] As background information: I require that that inference engines must support "deleting triples", which is often considered a mystical feature that must not be supported in satisfying performance by open-source inference engines. Especially, deleting facts must not cause a full re-computing of inference on a million triples.

The drawback was, that the implementation required massive amounts of memory and processing power, not to speak of the manpower invested.

7.3.1. Separating Storage for Resources and PIMO

We experienced that the separation of storage for data about native resources and ontologies caused friction.

- First and foremost: there was no efficient way to run queries across both the PIMO and the resource store. This is insofar bad, as it was not possible to answer questions such as "which members of the Rome project have sent me e-mails within the last week" in one query. There are implementations that can spread queries on multiple repositories, but we doubt that they are as performant as when running on one store directly.
- The resource store did technically not allow updating of items (only delete and insert was supported). For ontology alignment purposes, changing of crawled data may have been useful.
- The ontological data stored in the PIMO may also contain longer texts that were interesting for searching, so the PIMO as such could have been seen as a resource. The separation was artificial — things in the PIMO were also resources that should be included in search, etc. This caused workarounds in the implementation of the search service.

7.3.2. Crawling Considered Harmful

The approach of copying all data into an index for search has several drawbacks. First, the needed disk storage size increases (its effectively doubled), second the data has to be copied from the native applications to the index, which is a source of inconsistency errors. Third, the crawling and synchronisation of the index with the native applications takes up CPU time and main memory. Nevertheless, all desktop and web search engines work this way at the moment.

In [SS05a] we have shown that crawling adapters (and a simpler adapter type which we coined "CBD adapters) are faster to program and cheaper to maintain than complex SPARQL interfaces to existing data sources. Seen from this economical aspects (human programming effort is more expensive than disk space or CPU time), crawling the data sources gives a better trade-off.

We hope that the need for this replication of data in a search-index diminishes soon. As a rule of thumb, we say that crawling and replicating resources into an RDF index is needed as long as the web is not fully read-write enabled and important data sources do not expose SPARQL interfaces. For the future, we hope to that all applications in the personal knowledge workspace expose SPARQL interfaces, then crawling the data for indexing is obsolete. Once this happens, the replication of native resources is obsolete.

SPARQL is the end of crawling

Now that we said that the resource-store is soon obsolete—shouldn't we have avoided building it in the first place? No, it is needed to show the possibilities that RDF opens for desktop and web indexing. The usefulness of SPARQL can be shown by using it on this crawled data, and this demonstration may inspire application providers to add SPARQL to their applications.

7.3.3. Validating the Development Process with Test Data

The semantics of the PIMO language allow us to verify the integrity of the data. In normal RDF/S semantics, verification is not possible. For example, setting the domain of the property *knows* to the class *Person*, and then using this property on an instance *Rome Business Plan* of class *Document* creates, using RDF/S, the new information that the Document is also a Person. In the PIMO language, domain and range restrictions are used to validate the data.

The following rules describe what is validated in the PIMO, a formal description is in Appendix A.1:

- All relating properties need inverse properties.
- Check domain and range of relating and describing properties.
- Check domain and range for rdf:type statements.
- Cardinality restrictions (required properties, multiple or not). This is expressed using the vocabulary shipped with the Protege RDF editor.
- Rdfs:label is mandatory for instances of "Thing" and classes.
- Every resource that is used as subject or object of a triple has to have a rdf:type statement. This is a prerequisite for checking domains and ranges.

Above rules are checking semantic modelling errors, that are based on errors made by programmers or human users. This rules was used to check if the inference engine correctly created the closure of the model:

- All statements that have a predicate that has an inverse defined require another triple in the model representing the inverse statement.

The rules work only, when the language constructs and upper ontology are part of the model that is validated. For example, validating Paul's PIMO is only possible when the PIMO-Basic and PIMO-Upper is available to the inference engine, otherwise the definition of the basic classes and properties are missing.

Validation for code quality

The validation can be used to restrict updates to the data model in a way that only valid data can be stored into the database. Or, the model can be validated on a regular basis after the changes were made. In the gnowsis prototype, validation was activated during testing of the system, to verify that the software generates valid data in different situations. Based on Paul's PIMO (Section 5.1), and functional mockup versions of the services, the developers of gnowsis were able to test the services and applications using automated JUnit tests. In the tests, we ran the ontology validation after any manipulation of data by the services or applications. This allowed us to easily spot software components that create invalid data, after the components finished, the validator reported errors. Although not all errors were found this way, test driven development based on mockups and Paul's PIMO were a great speed-up factor when developing gnowsis.

Ontologies are also validated during import to the ontology store. Before validating a new ontology, it's import declarations have to be satisfied. The test begins by building a temporal ontology model, where first the ontology under test and then all imported ontologies are added. If an import cannot be satisfied, because the required ontology is not already part of the system, either the missing part could be fetched from the internet using the ontology identifier as URL, or the user can be prompted to import the missing part first. When all imports are satisfied, the new ontology under test is validated and added to the system. A common mistake at this point is to omit the PIMO-Basic and PIMO-Upper import declarations.

By using this strict testing of ontologies, conceptual errors show at an early stage. Strict usage of import-declarations makes dependencies between ontologies explicit, whereas current best practice in the RDF/S based semantic web community has many implicit imports that are often not leveraged.

7.3.4. Why Storing Data in a Centralized Repository and not in the Files Themselves

During development of the PIMO approach, a decision was to be made if annotations are stored inside the files themselves or in a central repository. From the information management perspective, storing annotations directly inside the file has the advantage that the metadata is kept together with the file, and sending the file via e-mail or moving it to an external harddrive does not detach it from its metadata. From the architecture perspective, a central store for the metadata has the advantage that searches across all files are possible, and also that bidirectional links can be managed.

To embed RDF in local files, we found several approaches ([Sau03], p22):

- Using Adobe XMP to embed RDF in PDF files.
- The RDF/A standard defines how to embed RDF in HTML files.
- Use file systems metadata extensions, ReiserFS 4, HFS and HFS+ for MacOs have data and resource forks in files, NTFS has streams for this. This way, a file can be accompanied with a second entity for storing metadata.
- The metadata-enabled filesystem "WinFS" by Microsoft.
- The METS standard[18].
- Stop using a local filesystem but provide access to files using the WebDAV standard, which allows annotations.

Each solution works well in a certain environment, HFS annotations work on MacOS but not on Windows, XMP works for PDF files but not for plaintext files. With every one, we came to

[18] `http://www.loc.gov/standards/mets/`

the conclusion that they are not generally applicable. If an operating system supports the attachment of metadata to files, what about resources hidden in files? Think of the many appointment resources that are hidden in one single iCalendar file, should they be annotated on file level or resource-inside-file level?

Additionally, bidirectional links are a central feature of the gnowsis and the only clean architecture for bidirectional links is a central storage for links. If file A contains a link to file B, the bidirectional requirement would lead to change both A and B when establishing a link.

The last suggestion was based on the idea of using a local WebDAV server. This possibility was evaluated intensively for gnowsis, as it allowed a very elegant way to embed the use of metadata into the system. An additional drive would show up in the file system, where folder's would not match physical file folders but metadata classifications, similar to the *TagFS*[BGSV06] approach. Folders represented categories and files could be found in more than one folder, similar to MIT´s Semantic File System. WebDav and gnowsis would be tightly coupled. We did not follow this approach in 2003 because of the high programming effort involved, but see it as a very promising way to add annotations to files and to allow multiple categories per file, similar to SemFS.

7.4. Summary

In each of the previous Sections about the implementation we have shown that the implementations are sufficient to realize the suggested services from the architecture (Section6.2). Each component was optimized to be a minimal implementation, which also caused changes in the PIMO model and the overall system architecture. For example, the identification methodology of PIMO, using Things as unique representations for information entities, originated from a performance optimization.

Each part of the Architecture is realized: each part of the architecture fits exactly into one implemented service. More than that, our implementation was fast enough to index large data structures and some of the designed architecture (i.e. the Aperture Data Wrapper) is **adopted by commercial software companies** (Section 7.2.4). Three people used the software in production to improve their own Personal Information Management **for three years**. The general architecture as defined in this thesis was later (in the NEPOMUK project) adopted by Sebastian Trüg for the KDE project and will be shipped to millions of users in the following years. Some of the services (i.e. the Personal Wiki Service) are not realized in KDE by the time of writing, but can be added later.

Having a working open-source prototype of the Semantic Desktop supported this thesis in four major ways:

- The implementation validated that the *software architecture* consisting of *PIMO ontology, services, and applications* is implementable and works as intended.
- The prototype was used in real-world experiments which will be shown in the next part of this thesis.

- The prototype was used by ourselves for years. By “eating our own dogfood” we learned what Semantic Desktop means and many new ideas for the implementation appeared while using the prototype.
- Other researchers did build on top of our work and discussed it.

In the next Part, our own evaluations are presented.

III

Discussion of the Paradigm

CHAPTER 8
Case Study: Usability Evaluation of Personal Semantic Wikis

> Facts are meaningless, you can use facts to prove anything that's remotely true!
> Facts, schmacks.
> Homer Simpson, Simpson Episode 5F05 [1]

In this part, the idea of *Personal Information Management on the Semantic Desktop* is evaluated using the software elements described in the last sections. In two different studies, we want to answer the question what advantages the Semantic Desktop architecture (ontologies, services, applications) provides beyond the state of the art. Two practical evaluations in which the Semantic Desktop has been deployed have been done. In the conclusion, the last part, we provide a clear argumentation how the architecture fulfils the requirements stated by the PIM community.

[1] `http://www.snpp.com/episodes/5F05`

8.1. Usability Evaluation of Personal Semantic Wikis

> "User acceptance is key, and we badly need projects which, with the help of human and social science, study the cooperation between humans and Semantic Desktops."
>
> Jean Rohmer [Roh05]

The approach of Personal Information Models and Personal Semantic Wikis was evaluated with end-users in 2006. This chapter contains the approach to this evaluation and the results. The evaluation as such was conducted by *Dominik Heim* under the author's supervision, using the software created by us and in cooperation with partners from the NEPOMUK project. Especially Rósa Gudjónsdóttir and Kicki Groth from the Royal Institute of Technology, KTH, Sweden, supporting us with guidelines how to run end-user evaluations. Our results have been published before in [SDvE+06, SGK+06] the results of the usability evaluation presented here were gathered and published before by Dominik Heim [Hei06] in his diploma thesis, and, in abbreviated and recompiled form, in our 2008 paper [SH08].

The presented Evaluation shows the usability of the implemented tool, giving an indirect validation of the underlying architecture and model. The goals of this evaluation are given in the next Section, followed by the choice of evaluation tools, and then the actual procedure. At the end of this chapter a conclusion sums up the results.

8.2. Goals for the Usability Evaluation

The first goal of the usability evaluation was to show that the Personal Information Model (PIMO), and multiple applications built with it, can improve Personal Information Management.

The system under evaluation is the gnowsis 0.9 prototype, as described in Chapter 7. The evaluated systems have been:

- The PIMO Thing Editor as described in Section 6.4.2.
- The browsing interfaces described in Section 6.4.3.
- The Personal Semantic Wiki as described in Section 6.4.4.
- The Drop-Box as described in Section 6.4.5.
- The Tagging plugins as described in Section 6.4.6.
- The Semantic Search as described in Section 6.4.7.

8.3. Choosing the Right Evaluation Method

The personal semantic wiki is in its nature similar to the Haystack systems, integrating various data sources for personal information management. Adar, Karger and Stein faced the following problems when evaluating Haystack [AKS99]:

> Because of its personal nature Haystack is hard to evaluate in large studies. The system requires seeding by a user and continued use for adaptation to occur. Additionally relevance in the context of personal information is highly subjective. Standard methods for judging IR systems require a standardized corpora where experts have judged relevance of documents to various queries. These evaluation methods, which are largely based on precision/recall curves, are hard to apply to system such as Haystack.

Quan evaluated Haystack in certain scenarios for his dissertation [Qua03], this was also our approach to evaluate gnowsis.

There are different approaches to evaluating software in Human Computer Interaction (HCI). One approach is to setup an experiment in a lab-environment, reducing factors that may influence the experiment and invite test users to use the system in a supervised way. Such evaluation is typically done in a short time span (one day, or a few hours), due to the fact the the participants simply cannot spent more time on it. The problem with such experiments is, that the long-term nature of Personal Information Management, including storing and retrieving tasks, cannot be observed in a realistic way. Others (Barreau and Nardi) conducted interviews with users about their existing habits. There exist alternative methods that allow conducting a user evaluation not only in a laboratory setting, but also at the users workplace and with only a minimal setup and low costs. Rowley pointed out that testing in the field makes it possible to "directly involve the development team and in-house domain experts" [Row94]. Kuniavsky [Kun03] highlights in his book on user observation the fact that the importance of an evaluation lays in verifying the usability of a system and getting new ideas throughout a development circle. This implies an iterative development process, where an evaluation is done each cycle, after improving the implementation of the system.

This is different from the waterfall model where the whole system was designed and planned at the beginning, putting the evaluation at the end. Morse wrote an article about evaluation methodologies where she stated that iterative development has become a well-accepted part of today's software development because of its already achieved positive results [Mor02]. This is further supported by the availability of Rapid Application Development (RAD) tools such as Visual Basic or Delphi, which speed up the creation of working software at an early stage.
For the research done by us, the desired features of the system were not know in an exact way a priori, therefore we worked with multiple prototypes and an iterative process.

An evaluation can be assigned to one of three possibilities: inquiry, inspection and testing. Regardless of which group they belong, they should always be guided by an evaluation plan that includes information about data, users, tasks and metrics [Mor02].

When some of these aspects are ignored, the results of the evaluation can vary:

- The selection of participants is a crucial step in evaluation. Choosing the wrong ones (e.g. not domain experts, not from the intended user group) can falsify the results.
- Collected data needs to be measures with a suitable metric.
- The way the questions are posed can influence the answer (e.g. Do you also like this feature? vs. How would you rate this feature?)
- In questionnaires, the answers (e.g. "People did not like feature A" vs. "People did not find the button to start feature A")

In related work [Qua03, Boa04] we find evaluations based on interviews with users, accompanied by implementing a prototype. For our own evaluation, we used a combination of the following techniques.

8.3.1. Questionnaires

Questionnaires are written lists of questions that are handed out to the users and differ from surveys insofar as they are written lists, not ad-hoc interviews. Hence they require more effort, but reflect the users' desire and hopes [Kun03]. Questionnaires can be used at any level of the development process, depending on the kind of questions asked.

8.3.2. Usability Test

Usability tests are structured interviews focused on certain features of a prototype [Kun03]. Representative users (a person from the target audience) are asked to successively perform authentic tasks using a prototype of the system.

In a paper about usability evaluation Redish et al. allude to the importance of knowing the audience and their objectives before planning and conducting a usability test. *Tasks* that are to be done by the users during the evaluation should conform to these standards[RBB+02]:

- feasible for testing
- described in terms of end goals
- adequate complex
- in a realistic sequence
- reasonable length

In a well-noted paper, Greenberg and Buxton [GB08] also questioned if the current practice of HCI evaluations is good for all cases. Their key argument is *the choice of evaluation methodology—if any—must arise from and be appropriate for the actual problem or research question under consideration*. *Eat-your-own-dogfood* is one of these possible methods.

Rowley suggests to develop a script that is divided into three parts: introduction / preliminary interview, the tasks and wrap-up. Thus nothing gets forgotten and all interviews are consistent [Row94].

During the evaluation, the users are told to vocalize all his thoughts and feelings during the evaluation. The resulting *think aloud* [LR94] comments, accumulated during the interaction with the product, are noted and/or taped and analysed (in terms of success, misunderstandings, mistakes and opinions) later on.

8.3.3. Logging Actual Use

Another way to learn how users interact with a program is to collect usage data automatically. This is done using software to record important user actions like keystrokes, mouse clicks, time spent, etc. As this can lead to a huge amount of data it is advisable to focus only on data that is useful for the evaluation. Due to the fact that this method comes along with some privacy and anonymity issues, it is important to inform the user exactly on what is logged and how the information will be used later on.

Although this evaluation method seems good results without too much effort, there are several problems attached. It gives the researcher an idea of "who asked for what when" [Row94], but does not show what the user is *doing*, what he *sees* and what he *feels*. Even if he clicks on a certain button it is not said that he understood what this button is going to do. So all the information gained from this method has to be handled very carefully because the can be easily misinterpreted.

8.3.4. Contextual Inquiry

Contextual Inquiry (also known as proactive field studies) is a structured field interviewing method that aims to fully understand the users working-environment [Kun03]. Among the evaluation methods it is more a discovery process than an evaluative one. The evaluator talks to the user while he uses the system in order to learning about the about the current system rather than testing it. The conversation about the system during the inquiry brings up important facts that would not be discovered during an observation. Thus this approach enables the evaluator to really know the user's experience [RBB+02].

Given these four methods of evaluation, the question is now how to use such methods in the context of evaluating a PIM application.

8.3.5. Problems with Evaluation in the Context of PIM

In the field of PIM, the main problem is the long-term nature of any activity. Storing and retrieving tasks cannot be observed in laboratory settings in a realistic way. For PIM, privacy concerns are a problem and also the stability of the software prototypes, as people will depend on the usefulness of the system when evaluating it.

Common usability test take a short length of time and focus only on the GUI of a program [Kun03]. Kelly [Kel06] wrote an article about the problems of existing evaluation methods in the context of PIM. Such an evaluation is of longer duration and aims at providing information about the possibilities and limitations of the software itself not only the GUI. An evaluation in PIM should aim at providing information about the usefulness[2] of the software itself, and not only the usability of the GUI.

Another difference lays in that fact that is it not recommended to force user doing a PIM evaluation with fictitious tasks, as PIM on one hand emerges at unpredictable times and on the other hand it is highly associated to user activities. This makes it impossible to create a fictitious PIM task the user is as much addicted to as if it is one of his real PIM tasks.
Furthermore Kelly showed that it is important for PIM evaluation to let the user work his or her users own dataset, because of the unique behaviour of a user who works with individual data collections, tools, environments, preferences, and contexts. The only similarities between all the different users are some kinds of "high-level PIM behaviour" [Kel06] like retrieving or organizing [Kel06].

In related work on PIM [Qua03, Boa04] we find that evaluations based on interviews with users, accompanied by implementing a prototype are common.

In terms of quality, Nielsen [Nie00] argues that five expert users are sufficient to discover 85% of the usability problems in an interface. Many insights about Microsoft's "MyLifeBits" [GBL+02] prototype were found with one very dedicated user, Gordon Bell, who is also co-author of the publications. *Eating-your-own-dogfood* can be a method that reveals much about the possibilities of a system, independent of the background of the users.

8.3.6. Separating User Interface from PIM Functionality

A problem in the evaluation is that the measured results may not indicate exactly what was evaluated. An negative answer to the question "Was the semantic search useful for you today?" can be interpreted in multiple ways: the semantic search function is not an improvement beyond existing search, or there is another problem. We have observed, that many factors influence highly the quality of the answers. For example, the negative feedback about search can have these reasons:

- The user never uses search engines and prefers a bookmarking tool.

[2] Distinguishing between usability and usefulness is stressed by [GB08].

- During this day, search was not needed (for example, when the user was travelling or in a meeting).
- The semantic search was used but not perceived as an improvement.
- The feature was used, but the user interface is not appealing.
- The feature was used, but the implementation is slow.
- The user understands something different with "semantic search". After asking the question differently, he may answer: "ah, you meant that function, yes that was useful".

To get more precise results, our goal was to split up the questions and tasks in a way that allows us to evaluate the approach, and not the implementation. One approach to achieve this was to use contextual interviews with the users, while they used the system. Another was to split up the application into several specialized, feature-specific user interfaces. Each application showed then a part of the available features.

8.3.7. Constraints with the Gnowsis Evaluation in Special

For the gnowsis evaluation, we first considered running a public test of the software, allowing people to download it from our homepage and evaluate it themselves, in their current work situation. Interviews would then be conducted via online questionnaires and some on-site. This decision was based on previous evaluations in 2005 and in early 2006.

In the beginning of the evaluation, we asked the participants at a scientific workshop to participate in the experiment, 15 agreed to take part. When the actual evaluation period started, only a small fraction of the initial volunteers was still available. Investigating the circumstances in more detail brought up **multiple issues and restrictions of the evaluation**:

- The software was at the time of evaluation not fully functional, buggy, and slow. Having the test users on-site allowed us to track those problems and fix them immediately.
- From a previous evaluation we learned that contextual interviews can give more interesting feedback than questionnaires filled out by the participants. Having the test-users on-site allowed us to do this, otherwise we would have needed to travel to the users to interview them, creating more expenses.
- The domain of applied use was easier to control (university surrounding) and the level of expertise of the users was more coherent, all users were knowledge workers.
- Being on-site also allowed us to motivate the users to take part in the long-term experiment, users that were not on-site did not give so much feedback.

8.4. Evaluation Procedure

The previous Section gave an analysis of the existing evaluation methods, the general problems of PIM evaluation, and our problems of gnowsis evaluation. In May 2006 we improved our evaluation plan regarding participants and methodology and decided on the selection of tools. Focus should be an evaluation with 8 volunteers on site.

In Table 8.1 the list of evaluated use cases is given. A use case encapsules one useful feature of the software, it is both a feature of the software and accompanied online documentation[3]. Most of the cases originated from designed use cases which were used for development – before implementation of gnowsis, these described in detail how a specific task should ideally be performed by the typical user. Some of these use case descriptions date back to the initial requirements document of [Sau03].

Each case was presented to the user at the beginning of the evaluation during the *initial usability test*, and questions about how the participant benefited from each case were asked during the final *contextual inquiry*.

The **chosen procedure and methodology of the evaluation** was:

Participants Participants are in-house domain experts, working at the same facility, see Section 8.5.

Expectation Questionnaire A questionnaire about expected benefits.

The *Expectation Questionnaire* covered questions about expected features and benefits from the Semantic Desktop system. Refer to Section 8.3.1 for questionnaires. The participants were shortly introduced to the idea of the project and the main features, and then answer a Likert scale[4] questionnaire about which benefits they expect from the software. The original questionnaires are given in Appendix A.4. The results of the questionnaire are listed in Section 8.7.1.

Initial Usability Test A Usability Test combined with an introduction of the system.

Immediately after installing the system followed a scenario-based *usability test* (see Section 8.3.2 about usability tests in general). Then first interactions with the user interface were explained by the interviewer and done by the user. Contrary to common usability tests, this one had to be conducted on the users local computer instead of a prepared workstation that offers the very same conditions to every single participant. It would have been less helpful to conduct an evaluation on another dataset than his own [Kel06].

A scenario consisting of various use cases (see Table 8.1) served as usability test and was used to familiarize the user with the software, giving a quick overview over its main

[3] The use cases are shown, including instruction videos, on this website: `http://gnowsis.opendfki.de/wiki/GnowsisUsage`

[4] A Likert scale is a psychometric scale commonly used in questionnaires. In our work, a slight adaptation of wording was done for the answers. `http://en.wikipedia.org/wiki/Likert_scale`

Nr.	Use Case name	Description
1	Installation	A wizard that leads the user through the whole installation process
2	Integration	Integrating already existing information into the PIMO (address books, flickr tags, etc.)
3	Ontology Browser	A user interface to see and edit ones personal ontology
4	Show Thing	A user interface to focus one one single instance of the PIMO
5	Annotating	Consists of three parts: a) Relating things using default relations b) Relating things using user-defined relations and c) Relating things with resources
6	Semantic Wiki	Using the semantic enforced wiki to make comments about the thing within the PIMO
7	Drop Box	Semi-automatic moving and classification of an unknown file (e.g. a paper just downloaded from the internet)
8	Tagging Emails	Relating an email to the PIMO
9	Search	Retrieve information from th PIMO; consisting of two parts: a) Quick search b) Browser based search

Table 8.1.: The Use Cases to Evaluate

functions and abilities. In Appendix A.3 you find the **step-by-step guide used by the researchers**.

The test helped us to separate the problems that arise form the user interface from the software architecture based problems – the participants learned to distinguish what behaviour of the software was due to bugs and what behaviour was a useful feature. Learning how to use the system effectively was an important step considering the following long term evaluation – it saved time needed to assist the users (we did not need a "help-desk"). Results from the initial usability test are given in Section 8.7.2.

Team Approach Weekly meetings where all participants exchanged success stories.

The good results of the initial user training encouraged us to use the *team approach* introduced by Morse [Mor02]. His idea was to conduct an evaluation during which the users exchange their experiences with the software. This procedure keeps the level of discussion about the software high and the shared knowledge is an advantage for all participants as well as for the evaluation results.

The minutes of these meetings were not recorded, the feedback was used to improve the

software and circulate success stories as a learning medium.

Intermediate Interviews Short individual interviews during the evaluation.

The whole evaluation was accompanied by short interviews from time to time answering user-questions and addressing problems with the software. This was needed to capture ideas to improve the software. Also we were able to identify major bugs in the software and fix them before they discourage the user from continuing the evaluation. Feedback during these interviews were recorded as tickets in the bug-tracking system, or the bugs were fixed immediately.

Activity Logger A statistical logging software that records usage statistics of the software.

Logging software was presented in Section 8.3.3. For gnowsis, the logging software was written by us from scratch and embedded into the software prototype. In important methods of the software, logging lines were added to the methods to measure when the method was called. Depending on a configurable log-level (which showed up in the installation dialog and was later changeable) the activity logger would including the text value of parameters. Although the logs were only kept on the participant's computer, it was possible to protect the privacy of our users. Also, the software was used in production by outside users not participating in the evaluation, for them it was important to switch off the logging[5]. The results of the activity logger were used to cross-validate the answers given in the final interviews and are discussed in Section 8.7.4.

Contextual Inquiry Final interviews at the workplace.

The evaluation is concluded with a final *contextual inquiry* (see Section 8.3.4 about contextual inquiry in general). In former evaluations we realized the importance of direct conversation with the participants. Interviews provide much more feedback than for example logging the actual use, because the user can verbalize his thought and impressions as well as his problems. Hence the evaluator can read between the lines and ask each user individual questions to collect information about the processes themselves as well as the consequences of the behaviour [Kel06], that could only hardly be extracted form a log file. This pieces of information include elaborate processes, occurred problems, suggestion for improvement, user contentment, etc. The inquiry consists of two parts. The first one was mainly based on the evaluation use cases introduces in table 8.1. The user has to rate each use case due to its importance and frequency of usage as well as the most often used features. The second part aims at the general statistics of the long term usage. We ask for information about the frequency of using the software as well as features the participant missed during his interaction.

One of the most important question to us was for what tasks and goals the software tools was used. Given such a generic tool as the Semantic Desktop, what problems will users solve with it? This gives us examples what users invented and to what direction we have

[5] Keep in mind that we were evaluating a publicly downloadable open source project with high security and privacy requirements of the users

to concentrate on while developing future prototypes. Results of the contextual inquiry are shown in Section 8.7.3.

Chronological Steps After the first interview, the following long term usage benefits as the participant already knows the most important features of the software. We also expanded the former planned long term evaluation for our new plan. Due to the fact that we no longer had the physical distance between evaluator and user, it was possible to provide support when errors occurred.

8.5. Participants

The participants came from our close vicinity, the DFKI GmbH and the Kaiserslautern University. A geographic and thematic closeness supported the study because participants were always available, the distances were short and, comparing to the external domain experts, they were interested in evaluating the software. Most important, these participants were available for the contextual inquiry interviews at the end and for the usability test at the beginning. Also, it was possible to fix software problems in-situ.

To get useful results, *11 volunteers* on site where then asked to participate, from which *8 participants* continued the whole evaluation. The expectation questionnaires are therefore based on the data of 11 people, whereas the user interface feedback and log file analysis is based on feedback from 8 participants. More background information about the selection of participants is given in [Hei06, p81]. The participants were not financially compensated for their effort.

All of the participants worked within DFKI, our company, two were from departments that are not related to Semantic Web, six were related to Semantic Web and our Knowledge Management Lab. This biased them to rate the prototype better than it actually was, but also let them be more forgiving when bugs and problems occurred. Their ages ranged from 25 to 40, one participant was female. All participants were familiar with desktop computers and general PIM activities. All participants but one were native German speakers, their feedback was translated to English by the authors of this paper.

Participant A was a male senior researcher who was also occupied with software project management and consulting. He installed gnowsis in early 2006 and was still using it in May 2008. He has experience in Semantic Web technologies and semantic modelling. He was using *gnowsis* often each day. His work documents include 8300 files within 1160 folders. He created 1196 elements in his PIMO. B was a female junior researcher engaged in writing a PhD thesis and research project work. She has been using *gnowsis* since July 2006 and was still using it at the time of writing. She has experience in Semantic Web technologies and semantic modelling. Her work documents include 75900 files and 11700 folders. She created 465 elements in her PIMO. Participant C and D were student workers of other departments who had programming experience but did not know about Semantic Web. The four remaining participants were researchers from the Knowledge Management department.

8.6. Conducting the Evaluation

The evaluation of the system was conducted by Dominik Heim, in the time of July and August 2006. In the first week of July, all users were visited in their own offices. The evaluation scenario A.3 was handed out to each participant. It is the guidance for *initial usability evaluation of installation and a tutorial tasks*. While they tried to accomplish the tasks described in the scenario on their freshly installed Semantic Desktop on their own PC, Dominik Heim took notes of what the participants said, both encountered problems and enthusiastic comments. Participants were instructed to *think aloud*.

The *initial usability evaluation* lasted for approximately one hour for each participant. After it, each was ready to start the *long-term evaluation period* of one month.

The *team approach* (described above) turned out to be a key element in the evaluation process, as users explained each other how to use the software and they shared their experiences, how to solve practical problems with the Semantic Desktop. In the long-term phase, Dominik Heim acted in a supporting role, and not inquiring about the system. We did ensure though, that technical problems did not restrain the users.

In parallel, the *activity logger* collected statistical data of which actions the user did with the system.

The end of the evaluation was the *final contextual inquiries* in the first week of August. The interviews were conducted in the office of the participant, at the knowledge workspace. While the user was working with the system, the interviewer addressed all topics of the interview guidelines. Emphasis of the inquiries was on possibilities and limitations of the Semantic Desktop regarding its use for PIM. Additionally, usability problems were indicated by the participants, which were not discovered during the first usability evaluation.

8.7. Results of the Usability Evaluation

Key results of the case-study are:

- The *drop-box* component increased productivity as it allows to file items faster than without the assistance.
- The possibility to add multiple categories to a document was used, in the mean 2.5 categories were attached to a file, which is significantly more than the single category a hierarchical file system provides.
- The *gnowsis desktop search* was used very frequently. Users answered in the questionnaire that they found unexpected information and that the categories provided by the PIMO helped during retrieval.
- The subjects stated, that the context-sensitive assistance came up with unexpected, surprising information items (e.g., documents) revealing new, useful, cross-references.
- The participants agreed that the PIMO reflects their personal mental models.

The results of our evaluation can be divided into three main parts:

1. Expectation Questionnaire
2. Graphical User Interface (GUI)
3. Contextual Inquiry about the Use Cases

Subsequently they will be listed in detail.

Note that this is not an evaluation based on representative subset of users. Each user pointed out what he especially liked and what he did not like at all based on their very own perception while interacting with our software.

8.7.1. Results from the Expectation Questionnaire

The questionnaire was composed of two parts. Part one, a *Likert scale* is shown in table 8.2. One out of ten statements was given (left column) and the participants had to specify their level of agreement from *highly* to *not at all* with the neutral option *I don't know* that should prevent the user from randomly selecting one option and thus causing bias for the evaluation. These results are given in percentage of users that ticked this answer.

Table 8.3 shows six follow-up questions, related to the topics asked before. The top answers are marked **bold**.

	highly	**much**	**some**	**rather few**	**not at all**	**I don't know**
Speed up my workflow	18%	9%	**55%**	9%	9%	-
Help me structure my documents	9%	**45%**	36%	9%	-	-
Help me organize my documents	18%	36%	**45%**	-	-	-
Provide additional information to resources	27%	**45%**	18%	-	-	9%
Be easy to use	**36%**	18%	18%	27%	-	-
Be quick to learn	9%	**27%**	18%	**27%**	9%	9%
No period of vocational adjustment	9%	-	**36%**	18%	9%	27%
Provides an intelligent (semantic) search functionality	**56%**	18%	27%	-	-	-
Permanent assistance	-	**36%**	27%	18%	9%	9%
Provide a possibility for cross-application linking of resources	**45%**	27%	18%	-	-	9%

Table 8.2.: Results from the Expectations Questionnaire (Part 1)

Nr.	Question	yes	no
1	Do you use a desktop search engine?	**55%**	45%
2	Do you contribute to a wiki regularly?	**73%**	27%
3	Do you use a wiki to organize yourself?	**64%**	36%
4	Do you use tools that help you to organize your (local) data (semi-) automatically?	**73%**	27%
5	Do you use the same structures in different applications (e.g. email client, filesystem)?	45%	**55%**
6	Do you use a blog-system to store personal information you won't find again otherwise (e.g. bookmarks, text notes)?	**55%**	45%

Table 8.3.: Results from the Expectations Questionnaire (Part 2)

8.7.2. Results Concerning the Graphical User Interface

Feedback about the user interface was gathered in verbal form during the usability test at the beginning and the contextual inquiry at the end. This feedback was categorized both into *positive and negative* and by which *application or service* was used. For this summary, only feedback given by one than more participant is listed, and *cosmetic feedback*[6] is omitted. *This positive and negative rating gives an entry point to specific feedback written in the next Section*. The positive feedback is the following:

- 63% of the users explicitly mentioned the *autocompletion feature* of the *semantic wiki application* very useful.
- 38% of the users mention the web-based *search* interface as useful because it gives a detailed overview on results. The possibility to see related items of search results was a key feature.
- 25% of the users found the drag'n'drop feature in the Swing user interface important to create relations.

Negative feedback is the following:

- 63% noted that it is not possible to filter the Miniquire view. This feature was added to the software during the evaluation.
- Half of the users noted a lack of online help, especially for the drop-box application.

[6]Cosmetic feedback is of aesthetic nature, such as "I want that button blue". If possible, such requests are implemented immediately or are passed on to the developers for later fixing, but are not interesting for this thesis.

- Half of the users experienced to lose their work context when switching between the Swing-based Miniquire application to the web-based semantic wiki.
- Half of the user had problems working with the vocabulary used by the software: "class, subclass, instance, thing, tag, resource".

The main point of critique (filtering in Miniquire) was possible to add during the evaluation. Lack of programming resources accounts for the missing online help and the context switches between web and desktop software. The problems with the semantic web terms remains a scientific question. For the Semantic Desktop, the intended solution is to acquaint users with the terminology by providing interfaces based on known metaphors: blogging, wiki, and tagging (as described in Section5.5). Switching the whole application into a limited "only tagging, blogging, and wiki" mode (and thus hiding the fact that classes and properties are used underneath) would help to reduce complexity for beginners while still using the PIMO model underneath.

8.7.3. Final Contextual Inquiry about the Use Cases

In this section we present a summary of the positive and negative feedback on individual use cases[7] acquired during the contextual inquiry at the end of the evaluation and the ad hoc interviews during the long term evaluation. More details are available in [Hei06, SH08].

Half of the users identified the *Thing Editor* (see Section 6.4.2) as a way to get a complete overview on one item. Although the wiki offered a similar overview, the interface in the editor was rated more intuitive. In the use case of *relating things* using the editor, feedback was diverse. One group of users (a third) stayed with the simple default relations, whereas another third clearly preferred to use specific property types and to create own properties. The latter was hindered by the bad support of the user interface for custom links. Various use cases were reported by the different users, showing how they adapted the system to their own needs: linking talks with invited persons, linking persons with important documents, relations from your own publications to the workshops and filesystem folders, etc.

The *Semantic Wiki* was primarily used as a semantic notepad. More than half of the users decided to use it for unstructured content that would be "too much effort" to be modelled in detail using the relations. The auto-completion feature was often used and also allowed a simple way to create semantic links using the semantic wiki syntax (faster to use than the Thing Editor). The *Drop Box* was frequently used by two users, who also connected Things with folders. Three users did not do the effort to connect their existing folder structures with Things, thus not interested in this feature. If there was an automatic linking between Things and folders, these users would be better supported. For searching, the web-based search interface was seldomly used, because it was not integrated directly with the Java-Swing application. When used, the feature to show related items on mouse-over was seen as crucial.

With *e-mail* annotation, only a quarter of the users tried the plugin (it was cumbersome to install) and all stopped to use it sooner or later. The reason for not using the e-mail tagging

[7] The use cases were introduced in table 8.1

feature was that this version of the prototype had problems opening e-mails from Miniquire due to software problems, and that not all users did use the supported Thunderbird and Outlook e-mail clients. Especially, a feature to filter e-mails by Things *within* the e-mail client was mentioned as missing.

New Use Cases Additionally some new use cases were brought up by the users during the course of the evaluation. They are introduced in table 8.4. The percentage of users that expressed the desire is contained in the left column and the use case description is to the right. The two rightmost columns are used define the use case either as belonging to GUI or to PIM or to both.

User	Use Case	GUI	PIM
37.5%	Configuration assistant on first system start-up	X	-
37.5%	Tagging websites with tags from PIMO	-	X
25%	Directly annotating resources, i.e. automatic creation of a new thing that is related with the selected resource and opening the wiki	X	X
25%	Move and classify email (drop box for emails)	-	X
12.5%	Personal blog	-	X
12.5%	Tagging of already existing files without moving	X	X
12.5%	PIMO as RSS feed	-	X

Table 8.4.: New Found Use Cases

Most Frequently Used Components Table 8.5 shows the primarily used components on the right and the percentage of users that mentioned this component. Multiple nominations were counted. The table is in descending order.

User	Component
87.5%	Miniquire (PIMO editing)
75%	wiki
25%	Thing editor (annotating)
12.5%	Drop Box

Table 8.5.: Most Frequently Used Components

Main Purpose of Usage The main usage purpose of the gnowsis Semantic Desktop and the integrated semantic wiki Kaukolu is illustrated in table 8.6. The left column contains the percentage of users that stated the purpose on the right. The table is grouped by statements concerning project management and general statements. Again, multiple nominations were counted.

User	Purpose
37.5%	Project management
25%	Explicitly no project management because there are too few features
75%	Managing stuff (events, conferences, contacts, papers and publications)
12.5%	To-do lists (wiki)
25%	Personal notepad (wiki)
12.5%	Support for writing papers
37.5%	I organize my knowledge: I document everything that I consider to be important (e.g. things I read, my ideas)

Table 8.6.: Main Purpose of Usage

8.7.4. General Statistics

Activity Logger Results from the *activity logger* showed that one user had the system running permanently, one user 3 hours per week, two users 2 hours, one user 1 hour, and one user 30 minutes per week. The rest of the activity log was used to see what features the participants used. It did not help us much to find out if the system supports PIM or not, but showed what the users did with the system. Users did extend the default PIMO ontology with custom classes, 11 classes in the mean with a mean derivation of 9. Thus, some users did create many classes while other none. Only half of the users created custom properties, and then not in a significant amount (less than five). Users did create instances though, altogether 371 instances, but with a mean derivation of 81 between users, so some were not very active at all.

Table 8.7 illustrates the period of use per week. The left column lists the percentage of user and the right one contains the time in minutes. The table is in descending order (from frequently used to less frequently used).

Figure 8.1 depicts the PIMO of a participant after the evaluation was finished. In Comparison to the initial PIMO upper classes (Section 5.4.12) one can clearly see the created structures reflect the users personal mental model.

Figure 8.1.: An Example PIMO

User	Minutes per week
12.5%	continously running in the background
12.5%	180
25%	120
25%	60
25%	30

Table 8.7.: Usage Frequency

8.8. Conclusions Derived from the Results

The results of our evaluation can be divided into two parts. The first part deals with the GUI problems and the second part addresses the problems that are relevant for Personal Information Management and thus for the future of the Semantic Desktop and the semantic wiki. All of them will be subsequently discussed.

8.8.1. Conclusions GUI

The majority of the problems was found in the components, which were most frequently and intensively used (Miniquire, wiki, Thing editor). This is not surprising, since frequently used components are permanently examined by the users. Usability and design attention should be especially paid to the frequently used components.

The installation wizard proved to be a real enhancement of the installation process. It could further be extended by an introduction to the Semantic Desktop on the first system start, as well as by some kind of auto-configuration, because all users had more or less the same problems while configuring the system at the beginning.

Many little improvements in the semantic search (e.g. “stop”-button), the semantic wiki (e.g. better visualization of the relations) and Miniquire (e.g. automatic reloading, a possibility to filter the PIMO tree) could help to let the software become more intuitive for the user.

Features like the semantic search, the semantic wiki and Miniquire (the PIMO visualization) showed their amenities, but also have capabilities left that need to be improved in the course of the next gnowsis prototype (e.g. filters for semantic results, undo functionality).

A problem that was noticeable independent from a particular components, is the fact that too little help and assistance is offered to the user. Users are faced with many new features and needs further information how to use them. This could be realized with a help button in the menu of each component that leads to a wiki page that describes the use of this component and illustrates it with an example. Another possible improvement would be to reduce the “jargon words” as much as possible, because 62.5% of all users were confused by all the new terms.

To emphasize is the fact that the GUI feature, which was rated best by the users, was the auto-completion of the new introduced semantic wiki. This is a first indication that the wiki feature

was well accepted by the users. Another important outcome is the fact that users felt interfered when forced to switch between the Java-application (e.g. Thing editor) and their browser (e.g. semantic search, wiki). Therefore, concentration on either a Java-application *or* a web GUI may be a better solution.

The main result of the evaluation in reference to the Graphical User Interface is the fact that it needs additional improvements. This is not very surprising as the interface is only prototypical yet. Because of the fact that GUI problems are not the main focus of this thesis, they have been reduced to a minimum. The next section discusses the results regarding PIM issues.

8.8.2. Conclusions PIM

This section discusses the evaluation results in reference to the field of Personal Information Management. Therefore we first analyse the most important use cases and following bring them all in a more general context.

Integration The expectations questionnaire showed that nearly half of the participants use the same structures within different applications. Therefore the evaluation showed that there is a highly interest to fill the PIMO with as much already existing data as possible to lower the effort to maintain the PIMO as well as minimize time needed for acclimatization. The endeavours in this field have to be further expanded.

Ontology Browser Miniquire, the ontology browser of the gnowsis system, is the most frequently used component. Therefore one can say that this kind of visualizing the Personal Information Model is adequate to the user. None of the participants had problems with the tree structure. Improvements should focus on making the browser more intuitive for the user and more suitable to PIM. 37.5% of the users showed the need of focusing on a certain (sub-)class. This indicates that while doing PIM one is very into a specific task and does not want to navigate through (sub-)classes of another context.

Show Thing Half of the users stated that the Thing dialog gives them a good overview over a certain instance, all its relations and properties as well as its wiki text. From the view of PIM this means that the ability of giving a detailed overview over a single item is of very importance in almost the same manner than showing the “big view” of concepts.

Annotating The annotating-behaviour of the users during the evaluation does neither focus on instances (things) nor on resources. Both, things and resource have been linked in equal shares. The difference in annotating lies in the used predicates. Providing more sophisticated class-defined default relations would be sufficient for half of the users. For instance a document, in difference to an event, has an author. Therefore the class “Document” and each subclass of it should provide the default relation “has author”, whereas the class “Event” and each subclass of it should provide the default relation “has attendee”.

25% of the user expressed the need of user-defined relation. Personal Information Management should always provide the user the possibility tu use his (personal) words. Hence this feature has to be improved.

Semantic Wiki The following results are important for the course of this thesis as well as for the continuance of the semantic wiki within gnowsis.

The expectation questionnaire already showed that the majority contribute to wikis as well as use them to organize themselves. This is also reflected during the evaluation. The majority of the participants used the wiki as personal notepad. Some of them only for entering unstructured text and some for entering text in a quick manner that was transferred to their PIMO later on. The auto-completion feature helped to generate semantic links in a convenient and quick way. For PIM this means that the personal semantic wiki complements the structured tools with a simple way to enter information. As the wiki was the second most used component and none of the users did not use it during the evaluation, it can be said that it fills the authoring gap and provides a different view of the users PIMO. The evaluation showed that the possible fields of application are manifold (documentation, comments on files, contact information to persons, to-do lists, notepad).

Drop Box The evaluation shows that the Drop Box feature split up the users in two groups. One who showed interest in it and another who were rather sceptical. As this is a very innovative feature we consider it as normal that users are "afraid of the novelty" at the beginning. Automatically associating existing folders with Things would help to acquaint users with this feature.

Perhaps it would be a good idea to reduce the functionality of the drop box to auto-classification and make the filing optional. At least 25% of the participants wanted to just classify a document and those 12.5% that were afraid of the auto-filing feature would perhaps become used to auto-classify and try the optional moving later on.

Tagging Emails Due to the fact that 25% of the participants expressed the desire to have a feature to move and classify emails and even 73% declared to use tools for (semi-) automatic organization, it is likely that there is a certain interest in the use of auto-classification for e-mails. Nevertheless, the current implementation does not address these desires. The software problems with integrating the e-mail clients are also an issue.

Search The quick search functionality was "misused" by half of the participants as replacement for the non-existing possibility to filter the view in Miniquire. In term of PIMO this feature was one of the most important ones from the expectations questionnaire and the browser based search was found to be useful by the majority of the users, but switching to the web-based view was experienced as cumbersome.

New Use Cases During the evaluation participants brought up several new use cases for gnowsis. On one hand this is an indication that gnowsis is not finished, on the other hand it shows that the architecture is indeed understood as an extensible platform for further applications. Some of the desired use case were already planned for evaluation but left aside for the evaluated prototype. For instance the personal blog use case. In difference to the wiki a blog has a temporal context and can be used for example for to-do list, as they are also only of temporal importance to the user. A need for this feature can also be found in the expectations questionnaire, where 55% of the participants agreed when asked if they use a blog to store personal information. Although the blog was implemented in the software (by using a standard JSP wiki plugin), it was not evaluated to limit the complexity. Another feature that was already planned for the next prototype was the possibility to annotate websites with tags form the user's PIMO.

Main Purpose Of Usage The purpose of usage reveals another interesting contrast. 37.5% of the participants stated to use gnowsis for project management, but 25% also stated explicitely *not* to use gnowsis for project managemant, as it did not provide sufficient features for them. This shows usability for the specific case for some users, but a clear need to integrate a professional project management tool into the Semantic Desktop for the demanding users.

The majority of the participants used gnowsis for event managing purposes (e.g. events, conferences). This is not surprising, as gnowsis is a tool for the management of personal information. This is approved by 37.5% of the users, who used gnowsis to organize their knowledge (including their ideas).

8.8.3. General Result of the Usability Case Study

We evaluated the gnowsis Semantic Desktop and the included semantic wiki Kaukolu to approve our assertions that such a system will improve PIM. Improvements were found in terms of efficiency, productivity, satisfaction or any set of similar qualities, as Morse defined in [Mor02]. It supports the user in both structuring their documents by providing a semantic search and the possibility for cross-application linking of resources. The integrated semantic wiki was able to fill the authoring gap and provides a possibility for entering unstructured as well as structured text. Therefore it can be said that the aims articulated in Section 6.1 were fulfilled. A part of the objectives stated in Section 1.3 were addressed in this chapter, the evaluations following in the next Chapters complete the discussion of the approach.

CHAPTER 9
Case Study: Evaluating long-term use of the Gnowsis Semantic Desktop for PIM

9.1. Introduction

The *challenge* of our field is that evaluations with real end-users are scarce, and especially there exist no long-term study of Semantic Desktop usage. In this chapter, a long-term evaluation of the *gnowsis* system is presented.

The *participants* were taken from the same group as the previous study. A contextual inquiry was done after 22 months with two users who continued using the system. The procedure and the results are presented, followed by a discussion. Results presented in this chapter were published before in [SH08]. You will learn **for what purposes** the Semantic Desktop was actually useful.

In the rest of this chapter, RDFS classes, instances, and properties defined in a user's PIMO are distinguished from text by underlining. It is a reference to hyperlinks and emphasizes the fact that each element points to an RDF resource and can be browsed by users.

9.1.1. Decisions for the Long-Term Gnowsis Evaluation

We agree with the view of Greenberg and Buxton shown in the last chapter and consider *eating your own dogfood* and long-term contextual inquiries as methods that help us to learn about Semantic Web technologies in PIM. Our research question is to see how our *PIMO ontology* and the *software prototype* were used by the participants for PIM tasks, namely filing and finding information[1]. Especially it is important to know how users file information, find and re-find information, and maintain their structures using the Semantic Desktop. Also the relation between mental models and the explicit PIMO structures is interesting. In the long-term study the usability of the software was not under evaluation.

9.1.2. Participants

Nearly two years after the first evaluation, three users were still available and kept using the system. They were interviewed in a contextual inquiry in April 2008. One of them did only sporadically use gnowsis for daily work and is excluded from the results. The participants A and B (described above in 8.5) remain, to keep their anonymity we further only speak of "one participant" in the male form. Both participants described themselves humorously as *nitpicking*

[1] These tasks are considered essential PIM activities in [JT07]

information keepers, one used the German term "*Strukturierungszwang*". Also, both are colleagues of the main author of this paper. They should be seen as "*eat your own dogfood*" users, that are biased positively towards the system and behave enthusiastic about it. Other users do not share this enthusiasm, only 2 of 8 have continued using the system voluntarily. Nevertheless, the behaviour the users show and the structures they created in their PIMO tell us something about the usefulness of the presented Ontologies, Services, and User Interfaces for PIM.

9.2. Procedure

In this chapter, the choice of procedure is not described as thoroughly as in the previous Chapter, the material presented there also applies here (please refer to Sections 8.3–8.3.6).

Contextual Inquiry

As *procedure* a contextual inquiry was chosen, as this method brought the most interesting results in the previous study. Instead of taking written notes a video of the interview was recorded. The contextual interviews started with setting up a video-camera for audio and screen recording and an introduction to the process. Basic questions about the participant were asked for warm-up. These were questions about name, gender, occupation, since when and how often they use gnowsis. Going from the warm-up into the contextual part was by the question *For what did you use gnowsis last?*. Then participants began showing their structures and telling about experiences. If needed, specific questions were asked about the PIMO classes and instances they created, whether they associated files with instances, their web bookmark keeping behaviour, usage of the wiki, and how they created associations and instances. Then, the interviewer asked participants to continue working on a task that they need to do at that moment anyway as part of their normal work. To help the interviewer, the questions were printed out on paper guides and checked off using a pen, the detailed answers later filled-in by analysing the video. **The paper interview guides are included** in Appendix A.5.

Tasks during Interview

The tasks chosen by the participants were:

- Filing two documents that were in a "to be filed later" folder. This folder is maintained by the dropbox application.
- Compiling information about a research project to be sent to a possible customer.
- Adding information about a new student worker and creating a task for him.

At the end of the inquiry, feedback about the interview process could be given and the interviewer asked the participants to provide a copy of the *Evaluation Logger* logfile that contains the activities the users have been doing. Altogether, each interview lasted for about two hours including a coffee break. The interviews were conducted by Leo Sauermann.

One participant interrupted the beginning of the interview to fix modelling errors in his PIMO. He noticed "this thing should be part of that" and changed the structures ad-hoc.

9.3. Key quotes

Before going into details, some quotes by the participants reveal how the users worked with the system.

- I seriously use gnowsis since it has the wiki.
- I use it to check the status of this project, what was decided when and who said what. Also to track whom I have told what.
- When I use gnowsis' search for this person X, I know exactly that I will find the telephone number. When I enter it in google desktop, then I get masses of things but it takes time until I find the telephone number there. In gnowsis I know exactly that there is a slot. From my feeling, I am quicker with gnowsis.
- I don't model the world, I only model what occurs. If $ConferenceX$ in 2005 was never important for me, I don't create it. (an instance for 2004 and 2006 existed).
- Here I have the possibility to gather things, so that they are somehow connected.
- PIMO is organized how you really think.
- When I later have to write about a topic, then I find it here. It's a totally different approach to information.

9.4. Results

PIM activities

The interviews were screened and transcribed to text, some open questions were asked two weeks after the interview to clarify facts. In the following observed usage patterns are presented and classified into filing, finding, and maintenance activities. This classification of activities is introduced in Section 2.3, based on [BN95] and the summaries in [JT07]. First observations about the PIMO structures created by the users are given.

Classes and Instances

The predefined classes of PIMO are [SvED07]: Group, Location, Building, City, Country, Room, Document, BlogNote, Contract, Organization, Person, Event, SocialEvent, Meeting, Task, Project, Topic.

Users extended them with the following classes: Application, Domain, Hardware, Book, Notes, Paper Collection, Presentation, Proceedings, Project Documents, Project Plan, Project Proposal, Survey, Paper, Story (war stories, usage stories), Tips Tricks, Diploma Thesis, Thesis, PhD Thesis (or Dissertation), Project Work (a document), Pro-Seminar, Department, Research Institute, External Project, DFKI Project, Private Project, Conference (an Event), Conference Series, Phone Call, Workshop, Work Package, Student.

There may be more custom-created classes (but they were not shown to us). In the activity log we find that one user created 10 classes, the other 31. The question is, why and when users create classes. One user said about the possible subclass of Organisation, Ministry: *For example, I could have created "Ministry". But the effort to create it without having a benefit for it was not worth it. ...I model when I think that I can use it. Like Research Company. When I had 2-3 research institutes amongst the companies the pressure was big enough — I created research institute and changed the class (of the existing instances).* The same case was with External Project. The participant started with the predefined Project assuming the semantics of "my own project" until faced with projects run by others. Then, another class was created for those. Both participants did not use the system for *private data*, although one created a class Private Project but did not create instances.

The class Location was scarcely used. Participants articulated no need for locations as they do not classify information by location. Upon further inquiry they both have used the Google-Maps integration of locations at the beginning, but the lack of support to automatically geo-code and create locations made it unattractive.

Also the possibility to create explicit Groups to collect similar things was not used, instead participants used to collect things by making them part Of another thing. One participant created 43 Meetings, the other 2, with the explanation *if the Outlook integration worked better, I would use it more*. Both participants used automated features of gnowsis to create Persons from address book entries. But at a later stage, this feature turned out to be buggy and was not used but maintained persons by hand. One participant entered telephone numbers, and e-mail addresses into the system. 101 and 154 persons were created. Both participants used Topics to classify things (59 and 201 instances). Especially people and documents were annotated with topics.

Instances In total the first participant created 288 instances with 92 of them having wiki-pages, the other 959 with 148 wiki-pages. Most instances were created in the classes Topic, Person (and subclasses of person), Document, and Organization.

Relations The system provides *basic relations* which were used extensively: part-of (used 566 and 63 times) with the inverse is-part-of, related (122, 193) has-topic (401, 78), and the inverse is-topic-of. Besides these generic relations, more precise relations were available but the user interface supports the basic relations better. Hence the basic relations were used more and interpreted depending on context, see Table 9.1. Over the longer period of two years, these structures

Domain	Predicate	Range	Interpretation
project	has part	person	Person is member of the project.
organization	has part	person	Person works for organization.
project	has part	workpackage	The workpackage is part of the project plan.
meeting	has part	person	Person attends the meeting.
topic	has part	document	Topic isTopicOf document.
project	has topic	topic	Project is about this topic.

Table 9.1.: Interpretation of relations

are getting blurred and unprecise. Although the participant had no problem finding elements

and navigating, he noticed that the structures are "wrong". An example was a final meeting FinalMeeting about a report Report for company Acme. A kickoff meeting Kickoff started the process. The structures were (in simplified N3):

```
Kickoff a Meeting;
  hasPart PersonA, PersonB, PersonC,
          PersonD, FinalMeeting;
  partOf  Report;
  related TopicA, TopicB, ReportX;
  occurrence fileA,
          another_report_about_topicA,
          interviewnotes.
FinalMeeting a Meeting;
  hasTopic Report;
  partOf Report.
```

When looking at these structures during the inquiry, the participant noted that the has-topic relation between FinalMeeting and Report is redundant and removed it. Later the participant said he created the relations over time, to navigate faster.

One participant had a problem when associating tasks to projects. The project had multiple topics, and when creating a task that was related to the project, he thought about adding the same topics to the task. Being not able to do "the right thing" stopped him from annotating further. At this point, rules and inferred knowledge would help the users. Also, more specific sub-properties of the generic properties are needed.

Filing Information

Following is behaviour specific to filing information, in the tasks the participants have chosen to work on during the inquiry. One participant chose to create a Person representing a new student worker. He created an instance of Student, added firstname, lastname, and fullname ("for the search"). A relation to a project was created and a folder on the hard-disk associated. The participant wished to enter the skype-id of the student, but was not able as the property was missing, and the user interface made creating properties complicated.

Another task was to create a new task for this student. An instance of Task was created, then opened and the previously created student selected as related. To express that the task is about a certain topic, two topics were assigned. The participant said, that the formal task description (a one page MS-Word document needed for the Human Resource department to track the employee's tasks within the organization) would then be added to the task as occurrence, and notes in the semantic wikipage of the task.

Participants used their filesystem and e-mail system in parallel with gnowsis. As e-mail was not integrated well (the plugins had many bugs and were de-activated by the users), participants did not annotate e-mails, but expressed the dire need for annotating e-mails. One participant used *gnowsis*' web-tagging tool to file websites. With files, the drop-box application saves time **Drop-Box**

and helps decision-making when filing, therefore it was used much. The quality of automated tag suggestions was described as very bad, participants complained that they often had to do the tag assignment by hand. The rate of files annotated with gnowsis varies from folder to folder and application. For example, one folder with scientific papers was annotated heavily, whereas others weren't. Compared to file keeping using folders and folder hierarchies, the possibility of multiple classification was both heavily used and expressed as very positive both for filing and for retrieval. Both used the Drop-Box extensively, 386 and 149 times.

Noting text in the *personal semantic wiki* proved to be a key feature. Each thing in the ontology can have a correlated wiki page. Participants used the wiki page for different purposes: to write short notes defining what the concept is, longer notes with copy-pasted quotation and text snippets or to write down meeting notes. The wiki was also used to create web link collections by copy-pasting URIs into the wiki text (by the user who did not use the *gnowsis* web-tagging tool). Both participants discovered many hidden features of the Semantic Wiki by reading the documentation and used them creatively.

One participant created an instance of Task called Todo. On its wiki-page, he used the option of Kaukoluwiki to include other pages, and sections of other pages. The participant then created todo-sections in various other pages and included them all in the master Todo page. For example, the section *Things to do* in the wiki page of ProjectX was integrated in the master todo page. The inclusion was never removed or maintained from the master page, included pages do not show up once they are empty. This system allowed the user to *gather all todos in one place*.

Finding and Reminding

Miniquire

Both participants used the miniquire visualization as main entry point to the system. It is possible to filter the tree using text-search. This feature keeps the spatial arrangement of information of miniquire but filters it. At the end of the inquiry, one participant noted that the miniquire search box is most important for him.

During the interviews, both participants did always use miniquire when opening a specific thing. Once a thing was open, the linked relations to other things were used to navigate. One participant described a certain PersonX as *entry point* to more data. It was an external project partner being responsible for a certain part of the project. The participant navigated from PersonX to find telephone notes, workpackages, and documents related to the project. It seems that once a certain path to information elements is followed, it gets trained and re-visited many times later. The preference to follow paths and navigate amongst items was already noticed by Barreau and Nardi [BN95], "*users prefer to be able to go to the correct location on the first try*". Using fulltext search was often used when navigation fails.

One participant mentioned to gather information about a person X before doing an important business phone call. The relations allowed to step from X to previously entered phone notes, and to the project.

Orienteering Behaviour

The behaviour of browsing to an element, examining it, and deciding where to go next was also described as *orienteering* by Teevan et al. in their CHI 2004 paper "The perfect search engine is not enough: a study of orienteering in directed search" [TAAK04]. On hindsight we

see strong similarities of our observations and their observations of user behaviour. Key learnings from their work on orienteering are:

> "Orienteering denotes a search behaviour in which people reach a particular information need through a series of small steps.
> ...
> The relatively small steps taken in orienteering also appeared to allow participants to maintain a sense of where they were, helping them to feel in control, to know they were travelling in the right direction with the ability to backtrack, and to feel certain they had fully explored the space when unable find what they were looking for."

The gnowsis participants both claimed that they only want to write down what is necessary and model what will be used – the feeling of control starts already in the authoring process. Gnowsis allows three major ways to take small steps in orienteering: navigating and filtering in miniquire, clicking semantic wiki links, and clicking "related" things in the PimoEditor view. Gnowsis supports going "back" by opening new things always in new windows (the old ones have to be closed). We assume that a feeling of *certainty* is supported by two facts: first because all structures were created manually and are known by the participants, second because additional to navigating and orienteering, gnowsis also supports fulltext-search (including the wiki and files).

When asked about *what would you do if gnowsis would be taken away from you*, the participant said that the missing text-search functionality would not be such a problem, but the structures and relations. He would not have *confidence* in himself when searching files, because he relies on the structures. This statement further supports the importance for orienteering in PIM.

Maintenance activities

In PIM research, maintenance activities are tasks to reorganize or think about information [JT07]. One effective example use of the system for maintenance was the preparation for a survey. The participant did come back from an interview, having taken notes as a text. Later, he had to deconstruct the interview into parts. This is described as a "creative step": to read the transcribed text and create relations from the interview to other things, such as the project or the customer. For example, the topic S was brought up in a meeting by a customer. In the notes, the topic was mentioned but the participant did not know about it. As the customer will likely mention it again, he decided to create $\underline{S}$ as a Topic and add some text to it. An internet search brought up the homepage of organization SU which works primarily in $\underline{S}$. The participant changed the type of $\underline{S}$ to Company and attached the homepage. Two documents describing S were related. The participant mentioned that S will be important for other projects also, that was the main reason to create it in his PIMO as an instance. If not for the possibility to relate it to future projects, he *would have just added some text about S in the wiki*.

Remodelling

Generally, participants needed some time to learn how to model effectively. One participant decided to start a completely new PIMO after four months of use, to clean up.

From the larger pool of available features, the two long-term users both only used a very limited set. We learn that these features help them to fulfil their day-to-day PIM. Namely, the miniquire view as an entry point, the wiki to keep notes (and search them), the relations between things, and the relations between things and files and folders are used.

Other tools

One participant used *QuickAid* which allows to open file folders on the harddisk using a key-combination. Once activated, the tool lets you search for folders by typing the name and shows the results immediately. It is faster than gnowsis, and the participant noticed "*If gnowsis could also do this, I would not need QuickAid anymore*".

One participant used *google desktop* search in combination with gnowsis. Also *Mindjet's Mindmanager* was used before to take notes during meetings, but later replaced with gnowsis.

One participant tried to create similar structures before gnowsis. For each folder, a text file was created with notes saying what is inside the folder, but this proved to be hard (sic: "you can forget this").

Perceived Structures versus Real structures

Verbal use of PIMO

Both participants knew their PIMO very well. During the contextual interview, they used the same terms verbally as written in their PIMO. Upon asked what a certain thing in the PIMO represents, participants gave verbal explanations that are very similar to the wiki texts they have written. This confirms the nature of "supplement to memory" we envisioned in the first definition of the Semantic Desktop [SBD05]. For both final participants, it was apparent that they can use the PIMO structures effectively, without always being modelled precise, correct, or exhaustive. They were able to find things effectively using the miniquire sidebar. But their *perceived mental model* of the PIMO structures differed from the *stored ontology*. Especially the relations written in PIMO are different from the mental models.

Mental Model and PIMO

One participant said: *If I have a topic, I always know whom to ask.* Upon inquiry if this was in the past used to contact people, the participant did first only remember the first name of a person and showed a wrong person, and later remembered the right success story. Another example: *This person, Donald X, when I was looking at his homepage, I added the topics he works on.* Looking into the data, the topics were not annotated to the person but to the papers published by Donald X.

9.5. Discussion

For PIM we can conclude that the combination of wiki, tagging system, and ontology is a good basis to the Semantic Desktop. The wiki was the second most used component and all of the users did use it during the evaluation.

Looking at the small number of custom-created classes and the even smaller amount of created properties shows that the granularity of semantic modelling is not so important when used for

personal information management. Users did remember where things can be found and how to navigate to them, and followed paths along "entry point" things. For the navigation to work, the nature of the relation (part-of, is-related, has-topic) is not relevant. A daunting hypothesis is, that for PIM, the only needed relation is has Tag expressing that two things are related.

In general, the approach of the users is to only model when it is necessary and needed later. Participants repeatedly said *I do not want to model the whole world*. Rather, the model is used to explicitly remember important things or facts. As a side-effect this also kept the system usable. A technical limitation of the user interface is that the performance gets worse when many thousand instances are modelled, and the miniquire tree-visualization would then be crowded too much.

Classes

From the created classes, we learn that they refine a specific PIMO class and not the generic Thing. Also, the low amount of created classes shows that classes as such are not such an important modelling tool. Removing the ability to create classes at all may remove one burden of the user to decide when to create a class. Also, the selection of classes identified by [LT06] as useful for PIM were affirmed by the structures created by our participants. Only geographic location was not used as much as expected.

A problem is, that *classes* cannot be annotated like instances can. A class cannot have relations to things, nor can they be annotated in the wiki or be used as Tags for occurrences. So classes are excluded from most editing functionalities. Users wanted to annotate classes to add meta-information about the class. Given the class Requirements Specification (a subclass of Document, instances are concrete requirements specification reports from customers), the user wanted to relate this class with a document that has instructions how to write a Requirements Specification. The same happened with Survey where there are documents about "doing surveys" in parallel to the instances of the class Survey as such.

The contextual inquiry also influenced the participant and the data as such. At the beginning of the interview, one participant noted that *"this should be part of that"* and modified his PIMO. The behaviour of users doing organization work during interviews was also noticed by Barreau and Nardi [BN95].

2006 study

To interpret the results of the current study, let us reconsider again the key results of the 2006 case-study (8.8):

- The *drop-box* component increased productivity as it allows to file items faster than without the assistance.
- The possibility to add multiple categories to a document was used, in the mean 2.5 categories were attached to a file, which is significantly more than the single category a hierarchical file system provides.
- The participants agreed that the PIMO reflects their personal mental models.
- The *gnowsis desktop search* was used very frequently and users found unexpected information.

2008 interviews

It is interesting to note that the two long-term users who were interviewed in 2008 did not use the desktop search frequently. The key patterns learned from the interview in 2008 are:

- The ontology tree-view presented in the miniquire user interface is the major entry point to the data. This confirms the results of [BN95] where users also browsed first and only when not finding elements used search.
- The semantic wiki feature is crucial. The text notes were used daily and for various tasks. Users found creative ways to realize task management and note-taking.
- The preferred interaction of both users is *orienteering*, reconfirming the findings of [TAAK04].
- Applications which are not yet integrated are described as a problem, and a hindrance to use the Semantic Desktop. Plugins for all desktop applications are needed to further support users.
- The PIMO structures enable the users to replicate their mental models using associations and multi-criterial classification [Den06b]. The ability to structure information helped users to creatively file their information and remember elements. They used the structures as entry point to their files.
- Allowing to relate topics with documents as occurrence, grounding occurrence, and is Topic of was a modelling problem. Later versions of the PIMO ontology clarified the semantics of these relations.

multi-criterial classification

When users start doing more annotations, then *ranking and filtering* of items is crucial for an effective visualization. Still, a manually created (or natural order) of the items is needed also for filtered items. The claim that multi-criterial classification [Den06b] is both feasible and the crucial next step towards working Semantic Desktops was further supported in this study.

CHAPTER 10
Case Study: Expectation Interviews and Need for a Semantic Desktop

10.1. Introduction

To value the usefulness of the Semantic Desktop to a realistic audience, another study was done in Fall 2008. The goal of this study was to find which parts of the Semantic Desktop are useful, any why. The PIMO framework can be verified by comparing it with real structures already created by users. For example, given the main classification criteria (classifying by location, project, or person), can these criteria be found in existing file systems of users? Also, the idea of using the same names for file-system folders and e-mail folders was observed already by [Boa04], is this also the case for today's users? Once the current PIM systems of users are investigated, users can be asked to judge if they see an improvement in the Semantic Desktop approach. In this study, first the current PIM systems of users are investigated, then the usefulness of the Semantic Desktop to improve these PIM systems is evaluated.

10.1.1. Participants

Participants for this study were 22 people doing *knowledge-intensive work* from the geographical area of Kaiserslautern. Seventeen males and five female participants were interviewed. For the remainder of this chapter, *all participants will be addressed in male form*, to not distinguish between female and male answers. The age was as following: 16 were between 19 and 25 years, two between 26 and 30 and the other four participants are aged between 30 and 40.

All participants had a higher education or were in management positions. Seventeen were students of the Technical University Kaiserslautern. These students were from various semesters, through a pre-selection we focused on higher-semester students, the mean semester was 5, with a standard deviation of 2.7. Five participants were working, one managing a three-person trading company, four working as architects and urban planners. One participant was interviewed with an early version of the questionnaire to test the procedure. Some answers were not available for this user, as we changed the questions slightly. Where available, the answers of this user are included, for some questions only 21 users participated.

Participants were interviewed using their own computers with their own data. If a participant had two computers (i.e. work and home), it was up to the participant to decide which computer to use for the interview. Twenty participants use the computer for work and private data, one only for work (an architect), one used his private computer for the interview. The computer skills of the users were high. Six participants defined themselves as experienced users, beyond that,

ten also used complex software such as Photoshop or CAD, five additionally had programming experience. The operating systems used by the participants are shown in Table 10.1.

Operating System	Count
Windows XP	12
Windows Vista	5
MacOS	2
Linux Gnome	2

Table 10.1.: Operating Systems

The time effort of the participants was compensated with 10 EUR per hour, each interview lasting around two hours resulted in about 20 EUR compensation per participant. Each participant was instructed about the procedure, the compensation, and legal aspects before the interview. All participants had to sign a consent form (see AppendixA.6, Figure A.8).

10.1.2. Procedure

Each interview was to be conducted at the workplace of the participant. Some students agreed to do interviews at our offices at DFKI, which is close to the university where they usually work with their laptops. The researcher and participant arranged a private two hour appointment beforehand and arrived with a laptop to record the interview, a printed consent form, the printed interview questions as backup if the laptop fails, a printed info-flyer about the evaluation, and sweets [1]. The sweets were to break the ice in the interview situation and were offered to the participant during the interviews. Additionally, the compensation money was given in cash after the interview and the participant had to fill in a confirmation of receipt for university accounting. The interviews were conducted by Leo Sauermann and Martin Klinkigt.

First Interview

The first interview was about the *current PIM practice* of the participant and the individual PIM characteristic of the participant. This part lasted on average 70 minutes and the answers were recorded using the researcher's laptop in an online form[2]. The first questionnaire is included in Appendix A.6.

Then, a prototype of the Semantic Desktop was installed on the participants own work computer. Here, the NEPOMUK PSEW version [3] of the Semantic Desktop was used, a later prototype than the software presented in Chapter 7. The services and PIMO model remain identical and therefore the software is equivalent to the other studies. The key features of the software are introduced by the researcher and shown to the participant. All participants learn the following operations:

[1] Chocolates and cookies were offered to all participants to ease the tense. Student participants who visited DFKI to do the interview were offered free coffee. Two participants explicitly mentioned the sweets as positive experience of the study.

[2] To capture the answers, the online software from questionpro.com was used because of its extensive statistical result possibilities.

[3] It is also open source and can be downloaded from `http://dev.nepomuk.semanticdesktop.org`

- Creating a Thing by drag-dropping a folder from the filesystem into the Semantic Desktop GUI. Creating a project was the default example, adapted for some users.
- Creating a Thing within the user interface.
- Creating a sub-class and creating a Thing of this new class.
- Annotating multiple file with several Things.
- Annotating multiple web bookmark with several Things.
- Relating multiple things.
- Browsing through the created relations and opening annotated files and web-bookmarks.

After each feature presentation, the participant has to re-organize one information element from his own data using NEPOMUK, using the feature himself. After this introduction, the participant has organized about 20 elements using the Semantic Desktop and PIMO. At the end, the participant has time to ask questions, try out the system freely, and verify that the software works as expected. The interviewer verifies that the participant has understood the software by asking questions like "What does the system do?" and "How can you annotate a file using the system?". All participants were confident that they understood the presented features and were able to answer the verification questions.

The focus of the evaluation is not on usability — problems with the user interface were explained to the user and assistance was given. This support was done to let users fully understand what the software can do, independent of user interface problems. It was important that users can judge the features of the system on a function, conceptual dimension, and are not biased by a negative GUI experience.

As an alternative, if the software could not be installed on the participant's personal computer, the researchers computer was used for this introduction of the software. This limits the evaluation partly, but does not affect the final results — key folder structures of the user are recreated on the prototype computer to have the same file environment. The software has features to integrate the e-mail system, these features are only explained but not tried out as the e-mail tagging software is unstable.

Second Interview

After learning about the Semantic Desktop, participants were interviewed a second time. The second interview concentrated on how they would change their current PIM system based on the Semantic Desktop technology and approach. Especially, questions were asked in which areas they see a benefit of the technology . The second questionnaire is included in Appendix A.6.

One of the last questions is about room for improvements or disturbing parts of the interview. Our procedure was described as professional and positive. Four participants wanted more time to play around freely with the software.

10.2. Results

For the analysis of the results, the written questionnaires were analysed. The first questionnaire had 88 questions with 66 discrete questions (either five-point Likert scale or a numerical value) and 22 free-text questions. The second questionnaire had 17 discrete and 12 free-text questions. The discrete data was analysed statistically. The analysis of the data was computed using Microsoft Excel and SPSS. The answers from the free-text questions were translated to English and used as additional material to explain the numerical results. Based on the small sample, analysis was focused on mean and standard deviation values, for correlation values the *Pearson coefficient*[Moo06] was used. Answers included in this thesis were, if needed, anonymised.

In the next Sections, the answers to the questionnaire are summarised to see where PIMO can be used and what features were rated interesting.

- In Section 10.2.1 the use of PIMO in different areas in the PSI is investigated.
- Section 10.2.2 lists upper PIMO classes that are found in a user's filesystem hierarchy.
- In Section 10.2.3 the possibilities of PIMO to link between items are highlighted.
- The circumstances that influence a users decision to switch to a Semantic Desktop are analysed in Section 10.2.4.
- Questions addressing the main added value of the approach are discussed in Section 10.2.5.
- Section 10.2.6 looks on the exchange of information in groups.

The key results of the Fall 2008 study are found in the discussion Section 10.3.

10.2.1. Areas in the PSI

For the study, the PSI of the users was partitioned into several areas. These areas were *files, e-mails, bookmarks, calendar*, and *address book*. Many questions of the study were addressing effects on these specific areas.

The first question about the areas was the *priority* of each area for the user: "how important is an improved access in the following areas?" 100 points were distributed amongst all areas. In Figure 10.1 the answer to this question is marked as *priority*. To interpret the areas with a measure besides priority, a monetary value was given to the areas in the question: "Assuming your data was lost due a hardware accident and you pay 100 EUR to get your data back? Please distribute the 100 EUR as if you have to pay for the recovery of the data." Answers to this question are marked with *worth* in Figure 10.1.

A high *priority* was given to files and e-mails, lower values to the other areas. Comparing the results from *priority* and *worth*, we see that the files actually do contain a lot of information of worth, as do e-mails and information about contacts. Evaluating the answer to the question "why" users rated bookmarks and appointments lower, the answer was that appointments and

bookmarks can be recovered with less cost — bookmarks can be recovered by searching again on the web and appointments can be recovered by asking the information from other participants. For the remainder of the discussion, *priority* is used to describe the *importance of an area*. This is due to the fact that the scenario of losing data is rare and uncommon and not interesting for this technical thesis compared to the daily task of accessing information.

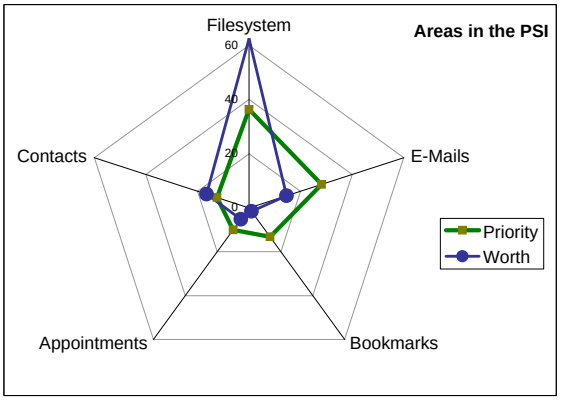


Figure 10.1.: Areas in the PSI: Access and Worth

Satisfaction Index in Areas For each area, the *satisfaction* of the user with the current solution was asked. This was done after each area was visited in other questions, so the users knew what each area meant. The question on satisfaction was: "With the possibilities of placing files into my filesystem I am satisfied...", the answer chosen from a five-point Likert scale expressing agreement. Four questions were asked for the areas filesystem, e-mail system, bookmark system, contacts [4]. The answers on the satisfaction level are shown in Figure 10.2, the mean values are between 3 and 4, 1 meaning dissatisfied and 5 full satisfaction

The standard deviation of the satisfaction answers to the filesystem was 0.83 whereas it was around 1.0 for the other areas. Combined with the high satisfaction with the filesystem, this expresses that most participants have found effective ways to organize their files with folders and are able to work effectively with these.

One user was not satisfied with the filesystem (rating it 2) while satisfied with e-mail (5), the explanation was this anecdote: *The user was an expert Linux user and knew how to use file-name search tools and e-mail search in his Thunderbird e-mail client. Both for e-mails and files, an elaborate hierarchical folder structure existed. He used many e-mail filter rules to automatically file e-mails into folders. When asked about his answers on satisfaction, he replied: "I am unhappy with the filesystem because I have an idea that it should be somehow possible to organize it much better but I have no idea how."*.

The two measures of satisfaction and importance can be viewed in combination. The following *Importance-Satisfaction-Index* is based on the approach of expectation interviews as sug-

[4] Organizing appointments was not measured, as the question would evoke reactions that are more relevant to the user interface of their calendar system and not to a categorisation system.

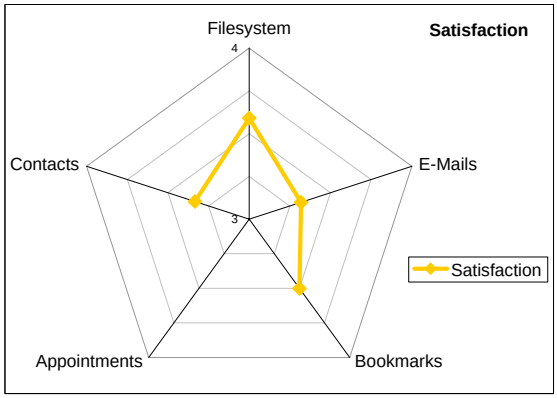


Figure 10.2.: Areas in the PSI: Satisfaction

gested by Andreas Dengel in [Den06a]. In Figure 10.3 the index is plotted, the *satisfaction* values on the X-axis with higher values to the right. The *importance* value is on the Y-axis with higher importance to the top. The individual answers of each participant for each area are plotted, with a slight distortion to separate overlapping answers.

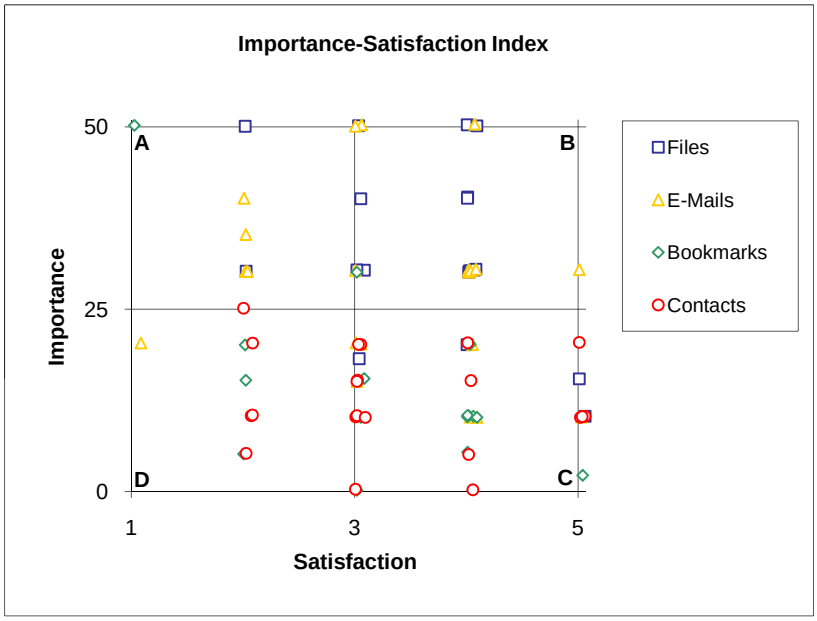


Figure 10.3.: Importantce-Satisfaction-Index

The analysis step is to find areas where satisfaction is low (showing a need to improve) and importance is high (showing that the benefit is worth the improvement), which is towards the corner marked “A” in the Figure. The mean values for files (importance: 36/satisfaction: 3.6) and e-mails (28/3.3) are towards that corner, but not in a clearly distinguished way. Looking at the verbal explanations of the users during the interview, we find that users are mostly satisfied

with their filesystem because they have created the structures themselves and, often after years of changing the structures, are now effectively filing and re-finding files.

Satisfaction Index After normalizing the importance I and satisfaction S values of each answer i into an interval between $0..1$, a satisfaction index F can be computed [Den06a]. The index is defined as:

$$F = \sum^{i} (I_i \cdot S_i) \tag{10.1}$$

In a similar way the importance can be multiplied with the "dis-satisfaction" of the users to compute a dissatisfaction index:

$$DF = \sum^{i} (I_i \cdot (1 - S_i)) \tag{10.2}$$

The index value for each area is given in Table 10.2. The values shown in the DF column correspond to the discussion of Area "A" in Figure 10.3, again we see that the filesystem area (with value 0.29) and e-mail (with 0.26) have a high potential for improvement.

Area	Satisfaction Index F	Dissatisfaction Index DF
Filesystem	0.71	0.29
E-Mails	0.51	0.26
Bookmarks	0.25	0.11
Contacts	0.23	0.11

Table 10.2.: Satisfaction Index Values

Answers on Benefit of Semantic Desktop in Areas Above, the hypothetical benefit of the Semantic Desktop was estimated based on current user's satisfaction and importance ratings. After having learned and used the Semantic Desktop themselves, the participants were asked in the second interview to estimate in which area they see a benefit of the Semantic Desktop.

The question was to rate the benefit of the Semantic Desktop compared to their existing system. The answers to this question were again on a five-point Likert scale, five expressing "high benefit". This time, an additional sixth question was asked to separate the benefit of annotating individual files (relating a file to multiple Things) from the benefit of annotating individual folders (annotating a folder as being the representative folder of a Thing). Both these features were learnt and used by the user in the experiment. The answers to these questions are shown in Figure 10.4. In the graph, the benefit value of folders is shown as a single point (*Benefit (F)*) on the filesystem axis. Additionally to the benefit as estimated by the users, the satisfaction values from the first interview are plotted in the graph.

For the benefit answers, the benefit of annotating folders was rated much lower than the other areas. The standard deviation of this value was 0.9, at the other answers it was 1.0-1.2. Users

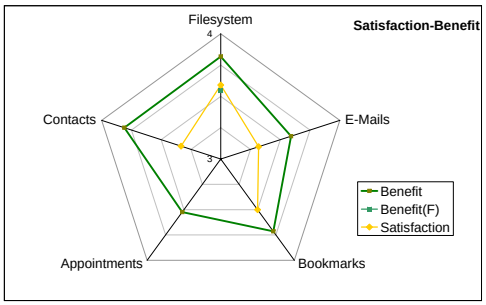


Figure 10.4.: Areas in the PSI: Satisfaction and Benefit

commented on this answer that they already encoded meaning in the folder structure and are content with it, or that a good folder is not needed any more when annotating with Semantic Desktop. Hence the added benefit of Semantic Desktop for folders is ranked lower.

Benefits within the e-mail area

In the area of e-mail usage, a significant relation can be found between the deepness of folder structures and the problem of searching for e-mails in the wrong folder. Whereas users always had deep folder structures for files, the organization schemes varied amongst the e-mail folder structures. Eleven participants had elaborate folder structures in their e-mails, whereas then had all their e-mails in one folder, leaving one participant with a few e-mails in a few folders. This correlates to the answers of failing folder search: "how often do you not find e-mails because looking in the wrong folder?". With high significance (error probability of 0.005 using a bi-variant correlation), a negative correlation between both exists. But no significant correlation exists between the e-mail organization scheme and the satisfaction with the existing e-mail. Users rated the benefits of the Semantic Desktop for managing e-mails positive (see Figure 10.4). Another significant correlation was between answers on the benefit of the Semantic Desktop for managing e-mails (in the second interview) and the answers to the question if a unified structure for e-mails, files, and bookmarks would be good in the first interview. The positive correlation (0.665 with an error probability of 0.001) between these answers suggests that users who want a unified folder structure amongst applications also see a benefit for organizing their e-mails with it (and vice-versa).

As immediate benefits for organizing e-mails, one participant using e-mail search frequently, being content with his e-mail system organized by many sub-folders noted to "immediately switch to Semantic Desktop if it supports e-mails". As use-case he suggested to connect specific work-related e-mail folders with specific work-related file folders. A professional noted that he would link important mails to projects, currently he prints them and files them into a physical folder. He would use the same organization then for his faxes, which are stored as image files on the computer.

10.2.2. Measurement of PIMO Upper Classes

The PIMO upper ontology, as described before in Section 5.4.12, defines a set of predefined upper classes to categorize *Information Elements* on the Semantic Desktop. The classes are generic and valid across domain and culture, and intentionally not very specific. In the first interview, two questions were about the existing names found in the existing folder hierarchy. If an existing folder maps to an upper class, the PIMO class is immediately useful for the participant, because the user already decided to use it for classification. If an existing folder does not map to an upper class, which upper class could be extended to represent the folder's contents conceptually?

Two set of questions were asked to gather data about the use of PIMO classes in existing folder hierarchies. Both addressed the folder structure within the "My Documents" folder of the user[5]. First, a *subjective view* was asked: "I use Time/Events/People/Projects/Organizations/Topics/Locations in my folder structure", the answers was to express agreement with this sentence on a five-point Likert scale. For each each area one answer was asked. The results of the answers to the subjective questions are shown in Figure 10.5. The results for *project*, *topic*, *time*, and *event* are higher than *organization*, *location*, and *person*.

Then, all folder names from the "My Documents" folder were extracted using a software tool and categorized by the user into categories: task, event, organization, location, person, project, topic, and "other". The last category was used as a fall-back if the user could not or did not want to classify the folder. Users had between 50 and 35000 folders in their "My Documents" folder, on average 4056 folders with a standard deviation of 8924. These numbers were due to folders created automatically by applications. For example iTunes created a lot of folders to organize MP3 files, or website authoring tools automatically created vast amounts of folders. To keep the study focused on the manually created, and important folders, we encouraged users to put auto-created folders into the category "other", which was then further removed from the analysis. Each folder categorized in this way to a PIMO class was counted as one usage of that class[6].

The aggregated results are shown in Figure 10.6. The class *topic* was used very often, also as a fallback class if the contents of the folder did not match other classes. *Event*, *project*, and *person* were in high use, *organization* and *location* less, and finally *task* almost never. In the diagram the high deviation values are clearly visible, these numbers will be interpreted by looking into the textual answers, in the following Section.

[5]The "My Documents" folder is subjective, in this study the term is used to identify the folder where the user "keeps his files organized". This can be one centralized folder or multiple folders.

[6]For extracting the folder names, a simple Java tool was programmed. For counting the lines, a Python script was used. Each was about 100 lines of code, both were run on each participant's computer. By running them on the computer and copying only the aggregated categorized sums to the result questionnaire, it was guaranteed never to have copied any information about the actual folder names to the researchers computer. This was to clearly show respect for the participants privacy and participants welcomed this and some expressed that they would have not allowed this study otherwise.

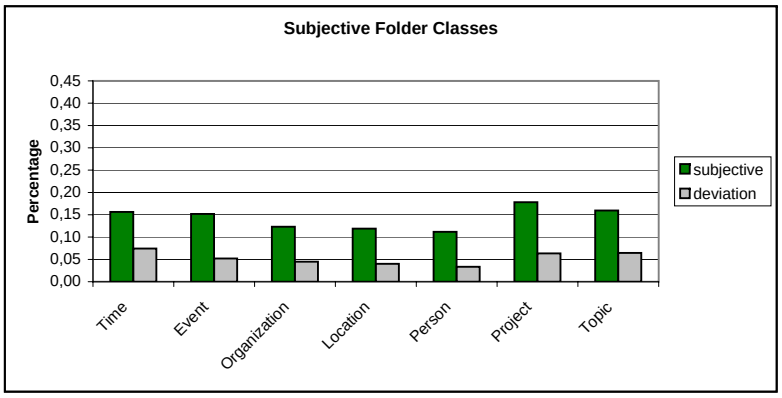


Figure 10.5.: Subjective Use of PIMO Classes in Folder Names

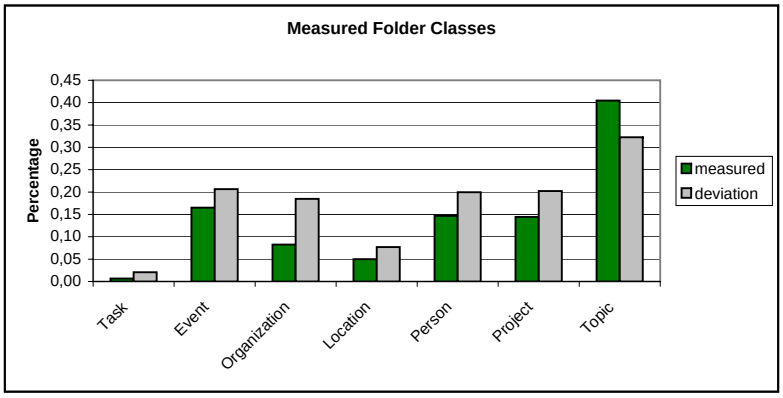


Figure 10.6.: Measured Use of PIMO Classes in Folder Names

PIMO Upper Classes Compared to Existing Categorization

In the previous Section, a subjective estimation, and a counted measure was given for use of the PIMO upper classes in existing file structures. The counted measure has a high standard deviation — which is due to the individual and personal categorization systems each participant developed. A problem of categorization appeared during the interviews, especially with the 17 students. The concept of "a course at University" was interpreted differently by each participant. Some classified each individual lecture as an Event and classified therefore courses as Events. Others classified the course as a project, others as Topic.

In the interview, participants were asked to explain their own organization scheme in a few words and distinguish which scheme they used for what task or area in their life. The first and foremost important pattern observed at seventeen participants' systems was for organizing *personal photo collections*. All of the seventeen participants had a primary scheme by time and event, and secondary schemes by place or person. Nearly all schemes for photos can be summed up in the simple formula for a photo folder:

```
/My Photos/<Date>[Place][Event][Person]

Example:
/My Photos/2007-Birthday-Mum
```

We found the same scheme both with students and in the architect's PSI. If a user had photos in his filesystem, they contributed to the events, location, and person numbers. Here, the PIMO classification into event, time, location, and person mapped well to existing structures.

The second pattern was found on the student's systems to organize university courses. Nearly all students had a top folder either called "study" or "university" or "uni". Below that, the lectures were either stored in the next level, or an intermediate structure was built based on a distinction between semester, year, area of study, or the status of "done / not done yet". As mentioned above, each participant had problems classifying the lectures. A lecture could equally well be represented as an event, a project, or a topic. After a participant decided how to classify the first lecture, the same decision was kept for the other lectures.

Third, all student participants (and some of the professionals) had major "areas of my life" in the main folder structure. These folders were typically named "private", "bank account", "telephone bills", "university", "the rest". Three participants explicitly mentioned a "the rest" folder, two explicitly mentioned a folder for "intermediate use" where downloaded files were placed before moved to a final destiny.

Some professionals developed a folder structure for planning projects that they used repeatedly as a template, this shows quite good how the stages of a project are handled:

```
01 project
02 input
03 research and literature
04 basics
05 design
```

```
06 output
07 final
```

These structures are not explicitly handled in PIMO and PIMO has no templating system to reuse the same folder structure amongst different projects. Using PIMO and the Semantic Desktop, the sub-folders could either be mapped to tasks or to individual stages, which are related to the project using a *hasPart* relation or sub-properties of that.

During the experiment with the Semantic Desktop software, the participants were instructed by the researchers to annotate existing structures using PIMO classes. This was straightforward and intuitive for projects, for cities, and for persons. It was not clear for lectures and university courses because participants again had to decide how to model them, either as projects, events, or topics. This stage in the hands-on part of the experiment was typically used to let participants create an explicit subclass for a "Lecture", or "Course". The decision on naming and the right superclass was up to the participant. After the class was created, the participants were confident to annotate multiple folders as being instances of the class.

10.2.3. PIMO To Relate and Tag

In the last section, the possibility to use PIMO classes was discussed. As shown in Section 5.5, PIMO can also be used as a tagging system to relate files and other information elements to Things. Besides that, things can be related to each other. Tagging in the filesystem and e-mails was an unknown feature to all participants[7]. To evaluate the usefulness of the PIMO possibilities, the questions had to be asked in a way that users can related to in today's terms.

In the first interview, one question was to judge the usefulness of a feature to add "more information to files, such as colours, icons, tags, or descriptions". Six answered with "highly useful", 10 with "useful", 5 with lower values, one undecided. Some participants answering "highly useful" were inquired, if they understood the question correctly. Two anecdotes were told about how participants tried to annotate folders by extra documents or other means that soon turned out too complex to manage on the long run — so the users did indeed understand the question and could map it to past experience.

A question was on a *unified stucture*, as already introduced in Section 2.5. The question was "Given you can manage e-mails, files, and bookmarks in the same structure, would it be easier to store and retrieve stuff?". Answers were given on a five-point Likert scale and were *positive*, 12 fully agreed, 5 agreed, 5 did not agree or were undecided.

To cross-validate this with the existing structure, participants were told to "hunt" for e-mail folders (and bookmark folders) that have the same name as a file folder. The names were written down. Once users did understand this, they were asked how many of their e-mail or bookmark folders would overlap with the file-folders. Answers were *negative*, two people reported that "many" names are equal, 3 people "some", 7 with "a few" and 10 with "no overlap". The last group included users who had no folders in their e-mail or bookmark system.

[7] One of the Mac users was surprised to learn about spotlight keywords. Despite the fact that he uses primarily spotlight to open files and not the "finder", he did not know about this feature.

To measure the possibility of *multi-criterial classification* [Den06b], a question was to rate the statment: “I could file my documents better if I could associate them to multiple folders.”. Answers were given on a five-point Likert scale, 6 fully agreed, 9 agreed, 2 undecided, 1 disagreed, 4 strongly disagreed. The reason for disagreement was explained verbally as “my folders are unambiguous, a file cannot belong into two folders”, “multiple folders for one file would puzzle me”.

The idea of relating multiple documents to each other was asked by verifying if the following situation happens to users often: “If you are working on a document for multiple days (please tell an example), do you open additional e-mails, files, or websites that you need to work on the document?”, The possible answers and the counted responses are given in Table 10.3.

Answer	Count
I never need additional documents	0
Sometimes, maybe unconscious	0
From time to time	1
Often I need related documents	12
Nearly always, and multiple documents	9

Table 10.3.: Need for Related Documents

Participants were asked to give precise examples to verify if they understood the question. To give some examples: students reported to work on “internship reports” or “course assignments we have to do in groups”, the professionals reported “project reports”, “design documents”, or “complex sales proposals”. Students identified related documents as “laws and regulations”, “course material”, professionals added “price lists”. After participants understood what a document and additional material are, the question was to rate the usefulness of a feature to connect both in a way that the additional material could be “clicked to open” when the document was open. Before answering, some participants asked “do you mean that it always automatically opens all related documents when I open a document?” which was described as negative. The interviewer then explained that the user would remain in control of this feature. Answers were *highly positive*, 12 participants rating this feature “highly useful”, 9 as “useful”, one as “not useful”.

After having used NEPOMUK to relate documents with Things (as explained in Section 6.4.2), three questions were asked in the second interview about these possibilities. The question “If the feature to relate documents to related material was implemented using PIMO, would this feature be useful?” was answered on a five-point Likert scale with 1 answer “negative”, 10 “positive” and 11 “highly positive”. In verbal feedback, participants described the linking feature with concrete examples, no user had problems using it and envisioning how this can be helpful. Compared to the classification problems of PIMO, the relation feature was described as easier.

In general, the possibility to add more annotations to files was appreciated. The idea to use a single folder hierarchy for both e-mails and file folders was not found in existing structure names but was found to be very useful by the majority of participants, in their verbal answers they

expressed the situation as "I would do e-mail and bookmark folders if they could be synchronized with the file folders". This confirms the observations done by Boardman [Boa04]. The concept of adding one file into two folders was not understood or deemed useful in the filesystem metaphor, but found useful when it was clearly separated from the filesystem while using the Semantic Desktop prototype. Users agreed that linking documents to related material as highly useful and also highly useful on the Semantic Desktop as implemented.

10.2.4. When Do Participants Reorganize?

Two questions were about a change of organization scheme. The majority of users (eight) re-organized "between four and five times", five re-organized "two to three times", seven for one time, and two participants "more than five times". All participants said that they reorganize (or at least optimize) their structures when they get a new computer or a new hard-disk. Two mentioned that they reorganized because they were unhappy with their existing file structure. Only one person reorganizes information in regular intervals (yearly) to clean up. One reorganized after the first semester at university, because he realized that the structures must be changed to be useful.

On the question, "why" they reorganized instead of keeping the existing structures, five people answered to increase speed when searching for things, three mentioned to remove unnecessary data, one mentioned that through reorganizing the situation worsened because he stopped in the middle and did not reorganize everything. Four mentioned that reorganization helped to get a "clear arrangement" ("Übersichtlichkeit").

One user mentioned that he does not reorganize any more because he mainly accesses files via the Apple Spotlight search engine, he was also the only participant who depended on the text search engine to open files (on the question of search, he answered that he opens files in 70 out of 100 times via spotlight, the mean answer was 13 times).

Reorganization for Semantic Desktop Introducing a Semantic Desktop environment for file organization would also mean a reorganization, to effectively use the system the user has to be willing to use it for categorization and retrieval. In the second interview, the question of switching to Semantic Desktop was asked. Given that a new computer would be shipped either with or without Semantic Desktop capabilities, 19 participants said to prefer a Semantic Desktop enabled computer, two users were undecided. The same question was posed differently by asking "you switched your organization system in the past, would you switch to Semantic Desktop today?"[8]. Ten users (45%) said they would switch if the system had more features, four if "others also use it", seven would use the system as-is. One would not switch soon because he prefers matured software.

Eleven users would annotate 50% of their files initially, nine users said to annotate more "to get most out of it". Two users said they would start with 25% and then annotate more files when

[8] Given that the system would work limited to the presented features, but more user friendly, running stable and fast — comparable to the current KDE 4.2 version.

necessary, starting with current projects. These general positive results reinforce the results on usefulness of the Semantic Desktop.

10.2.5. Main Added Value

In the second interview, multiple questions were addressing the "added value" of the Semantic Desktop in comparison to the existing system, and the "main problems" when switching to the system.

Generally, participants were answering positive on the features of the Semantic Desktop. On a scale of 1 to 5, (five being the best), feedback about improvement was 3.7 (see Section 10.2.1). Interesting from these answers to identify the added value is that we differentiated between "benefit to organize folders" and "benefit to organize files"[9]. Improvement was rated as 3.82 for files, with a standard deviation of 1.1. For folders, it was 3.55 (only appointments were rated lower with 3.52) with a standard deviation of 0.9. This was the lowest standard deviation, indicating that the use for folders is indeed low.

On the question of which structures could be imported well, answers were given as free text. In Table 10.4, participants are counted who mention an individual area.

Existing Structure to Import	No. Participants
Persons and Contacts	9
University related Items (Lectures, Seminars, Topics, Diploma Thesis)	8
Files	6
E-Mails	5
Bookmarks	4
Appointments and Events	4
Projects	4
Photos	3
Music and Multimedia	2
Folders	1
Applications	1

Table 10.4.: Structures with High Value for Import

Special use of relating Persons Especially we noticed that the possibility of adding *relations between people and other entities* (e-mails, projects, lectures, and documents) was mentioned by 8 participants. This observation is confirmed by the subjective view on categorization — the "person" category was ranked lowest with a mean of 2.14. In the objective view, on folder structures, persons did appear though (17% of the folders represented persons, 19% projects).

[9] This was also experienced during the demo by the participants: PIMO can be used to annotate a file or a folder.

Thus, the categorization by person is currently hidden in subfolders and complex structures, because the primary structures are usually topics or projects. With PIMO it would now be possible to use persons as additional classification concept.

10.2.6. Social Exchange

The participants were asked how their own folder structure relates to the folder structure of colleagues. This helps to indicate if domain ontologies (see Section 5.6) can be used and in what circumstances. The first question was whether the organisation where the participants work (or the university) has provided default folder structures to be reused. Overall, sixteen participants did not receive folder structures, five participants were provided with folder structures and used them for parts of their file organization scheme. Alternative answers ("received folder structures but are not used", "all folders are proposed") were not chosen.

More than 80% of the participants think that many folders are the same within their own system and on the computers of colleagues. Included in those, 20% think that half their folders are overlapping with colleagues. In the students case, all students had representations of their university lectures as folders, which will overlap with colleagues. The professionals often used network shares to organize data. One student lost all data once due to a hardware fault and copied the data from a colleague. The copied structures were not adopted, but left "as-is" in a separate folder.

Another question addressed discussions about folder structures. The assumption was that if people discuss about folders, they are likely to also discuss about PIMO structures and domain ontologies. Eight participants reported to *never* talk about folder structures with peers, nine reported *seldom*, five reported *often*. In the second part of the interview, 12 users agreed that exchanging information in the form of PIMO structures would help them, six answered "would help very much", four did not agree. The next question was whether they would "talk about PIMO structures to exchange experiences", 14 agreed, 4 very much. In both questions, positive answers were over 75%.

Verbal feedback about group ontologies often included the wish to stay with individual structures, but to adopt useful structures on a case-by-case basis. In the first interview, two students reported that they exchange information about their folder structures when working in teams on projects. One student worker and two professionals reported that their company has proposed a folder scheme layout such as "2008-11-28-ProjectName".

In the second interview, several mentioned to adopt the structures offered by the group — if they can decide which structures to adopt on an individual basis. One mentioned he would talk about structures if it was about group topics, like mutual projects, but not on private structures. Two mentioned the benefit that other employees can navigate in their structures if they reuse the same structures. We conclude that users see the primary benefit in adopting structures for their own benefit.

Partitioning the participants into groups: talkers and non-talkers All participants can be partitioned into *non-talkers* "people who never talked about their folder structures

before" (8) and *talkers* "people who did talk about their folder structures" (14) with others, we can analyse the answers to the questions if they would talk about PIMO structures. Both groups would talk about PIMO structures (mean value 4.79 for talkers and 4.5 for non talkers) and both groups would equally accept PIMO structures from others (5.0 for talkers, 4.88 for non-talkers). All standard deviations are around 1. Thus we can conclude that having PIMO structures available may support the group discussion process.

10.3. Discussion

The results from the individual areas in a users Personal Space of Information (PSI) show that users depend heavily on their files and e-mails for daily work. They have created ad-hoc systems to manage these information elements using their existing software, often not knowing about all features of the software. For the area of e-mail management, participants who saw a benefit of the Semantic Desktop in this area also saw a benefit in a unified folder structure across application borders.

The benefit of the Semantic Desktop is rated high for the area of contact management. The participants did not use their address book applications to categorize contacts or did not keep their contacts in electronic address books. With the possibility to annotate people using PIMO and relating them to projects, files, and folders, this would improve the current situation significantly. The possibility to annotate and relate persons was often mentioned in the verbal answers as the main added value of the Semantic Desktop.

The PIMO modelling concepts were used intuitively for concepts such as projects and persons during the observed hands-on session. When modelling elements that were not captured in PIMO yet, participants had to do decisions how to categorize new elements. This problem occurred also when categorizing existing folders during the first interview (without knowing about PIMO). Especially categorizing a "university course" was a challenge, because the course *is* at the same time a project (it starts and ends and has a goal), a topic (the scientific topic taught), and an event (the individual lectures). Once the decision has been made for one course, the other courses could easily be organized, like in the filesystem before.

The modelling decisions can be simplified by sharing existing models, which the participants did before the study and welcomed as a feature to help them model on the Semantic Desktop. All participants insisted to fully control their own model and asked if the Semantic Desktop allowed them to decide if structures from other people could be integrated partially.

In this study, we have shown that the Semantic Desktop approach indeed matches a need found in common desktop usage scenarios. Based on the answers, it can be concluded that in the area of file-keeping an immediate use exists, and in the area of web-bookmark management, the Semantic Desktop would make a considerable difference compared to the existing folder-based bookmark management systems. The possibility to link and annotate people was identified as a key improvement over the state of the art. We conclude that there is indeed a need for a Semantic Desktop in the software environments of the participants, and that the developed ontologies, services, and applications can cater for this need.

CHAPTER 11
Case Study: Increasing Search Quality in Proposal Management

In this chapter we describe an evaluation of the gnowsis Semantic Desktop in combination with the Brainfiler text classifier for proposal management at Siemens Business Services. The core idea is to support peer-to-peer search on the Semantic Desktop and to personalize the search results based on the organizational role of the user. Technically, this is implemented as a rule-based extension of the search service, as described in Section 6.2.4.

The evaluation was planned and carried out by Mark Siebert at *Siemens Business Services* (SBS) Munich, a part of the Siemens group. The department in question is performing consulting services for customers in the IT domain [1]. The study was carried out as part of the EPOS project, Mark Siebert was assisted by Pierre Smits and myself, with essential input from Heiko Maus who was project manager. It was published by us in [SSSD06]. My contribution is the PIMO theory, the Semantic Desktop architecture, and the implementation of the rule-based search engine.

The results of this study show an increase in search result quality compared to existing systems. It is included here to show how the Semantic Desktop approach can be adapted for a concrete business scenario using rules and a mid-level domain ontology. The results of this evaluation show in which scenarios it increases knowledge worker's productivity.

Knowledge Work Scenario

SBS produces many thousands proposals a year for its whole service portfolio, ranging from outsourcing to solution design and system integration projects. Therein sales teams consist of different roles, like proposal managers and sales managers, with different backgrounds, expert levels and functional tasks. Sales managers feed the first rough information, like request for proposal, together with the approach (e.g. price, competitive environment, etc.) into the proposal factories to gain a first draft story—like a management summary. The result is based on existing information and references, leading to open topics and requirements for further refinement.

Proposal managers work with organizationally pre-designed and re-usable content structures (see Figure 11.1, left) in a central repository. Sales managers often work in individual settings (see Figure 11.1, right) according their customer requirements and area of responsibility (e.g. sales requirements from public sector are different from e.g. software business) mostly on their desktop. Nevertheless, both work on same or similar documents, value propositions and rely on similar information pools (e.g. data stored in LiveLink) with about ten thousand knowledge resources.

[1] In 2007, after the evaluation, SBS was merged with other companies forming *Siemens IT Solutions and Services*.

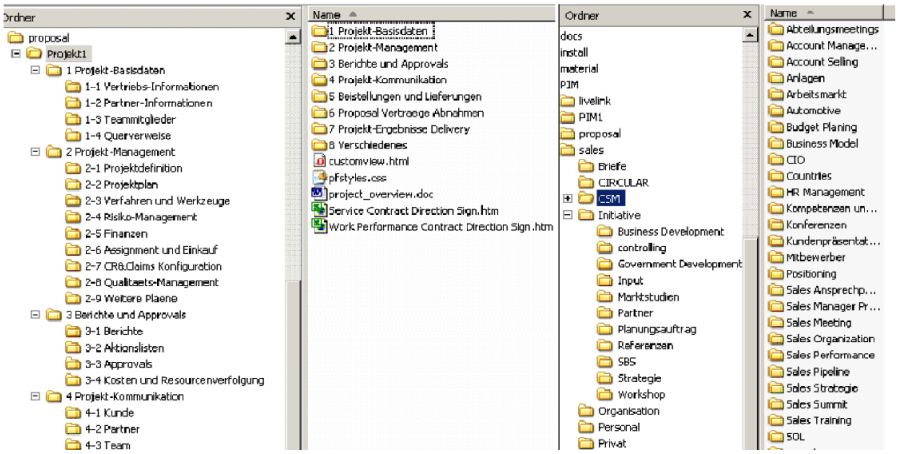


Figure 11.1.: Folder structure of proposal manager (left) and sales manager (right)

11.1. Goals for Proposal Management

Quicker response times to proposal requests and less production costs for standard proposals without quality reduction are current CRM requirements within increasing competition and market dynamics in the IT service market. Responding to customer requirements sales managers today either search for similar, existing, and successful standard proposals or ask an assistant to come up with a good draft. Other than proposals in product business, service proposals require a value proposition derived from and designed to the individual customer needs rather a value proposition of the product characteristics.

Existing knowledge management tools or *Proposal Automation Tools* already support general functionality like document handling and proposal generation. They lack deeper process integration, higher quality of search and respect of the individual characteristics of knowledge. With its new on-demand CRM platform SAP [SAP05] provides a virtual sales assistant guiding the user through the steps within the sales process (e.g. creating a value proposition or analysing competitor's products) and offering personalized information (like my reports, contacts, appointments, tasks). This provides a sufficient process integration and personalization of the interface but does not support the knowledge creation (e.g. a proposal) sufficiently. Sales assistants require extensive search capabilities to manage different information sources and to translate the customer requirements into searchable key words.

Interviews with SBS practitioners name the following reasons

- Heterogeneous storage paths with wording differences between peers (e.g. Sales Managers use customer—and Proposal Managers organizational language),
- Inexistence of knowledge assets in the central knowledge base and insufficient meta-data (kept on local desktop due to high publishing efforts),
- Insufficient dialog between roles due to communication hurdles misses respect of different

perspectives, increases proposal risks and lowers quality.

- Roles are only designed from a process-related point of view defining tasks. Intentions and backgrounds (expert level) are not respected.

Siebert defined the *Knowledge creation framework (KCF)* [Sie05] in 2005, a base for understanding the knowledge creation process as a combination of perspective taking and making. It describes six steps to develop knowledge assets (e.g. a management summary of a proposal) from an individual point of view - from gathering and mapping search requirements based on full-text, over classifying and categorizing them through ontologies up to consistent alignment through inferences, content selection, verification and production. In the evaluation, our prototype is used to explore the possibilities of semantic search and peer-to-peer technology, focussing on the steps of receiving and interpreting data.

The hypothesis under evaluation is that the process of search can be improved by increasing precision and recall of search results, and that the Semantic Desktop allows to integrate knowledge from various sources needed by the different user groups.

11.2. Software under Evaluation

Gnowsis was enhanced with the text categorization and retrieval engine BrainFiler [AG][2] providing the index service for all clients and determining structure and concepts of the indexed files. It creates term vectors of each file and cluster documents initially with their respective origin folder [ABM+00] to improve the precision of a search result. Information can be retrieved through fulltext search, semi-automated annotations, or manual annotations expressed in the *Personal Information Model (PIMO)*. These views lead to different perception of identical content, duplicated information in different categories and folders. On top of the annotations, the view on information is further influenced by the rules defined in the Semantic Search service (see Section 6.2.4). The fulltext and category search of BrainFiler is enhanced by gnowsis rules.

Deriving from the SBS proposal requirements, we identified four simple rules:

- If a project was found, determine project leader and author as a contact.
- If an entry, fitting the current customer requirements, was found, determine the relevant project(s) as a good reference.
- If an entry, fitting the currently required solution, was found, determine the referenced project.
- If a project was found, determine the project documents as a possibly good source document for the current task.

Those rules are stored in gnowsis using forward-chaining rules:

[2] Brainfiler is a follow-up product of Profiler, which was introduced in Section 2.4.

```
# Example for retrieving the project manager
# as expert contact
(?hit retrieve:item ?project),
(?project rdf:type org:Project) ->
querySparql('
CONSTRUCT {
 ?x1 org:HasProjectmanager ?m.
 ?m rdfs:label ?labelm.
 ?m rdf:type ?typem.
 ?x1 retrieve:relateHitTo _:hit .
 _:hit rdf:type retrieve:InferedHit .
 _:hit retrieve:item ?m .
 _:hit retrieve:textSnippet \'Projektleiter\'.
} WHERE {
 GRAPH ?g {
  ?x1 org:HasProjectmanager ?m.
  ?m rdfs:label ?labelm.
  ?m rdf:type ?typem.
 }
}
', ?project).
```

In SBS practice usually the simple rules appear context-related in combination, like: "If current role is sales manager and current task is proposal development and the found document is stored on a desktop of, or written by, a user identified as an expert about the searched topic, determine further details (e.g. assigned project, source documents and co-authors) about the document".

A full implementation in practice would have to model those combinations based on the PIMO ontologies and classes. Therefore, for each class and ontology its relation to others has to be predefined (e.g. expert-level .. show only certain document types).

The gnowsis web interface, described in Section 6.4.7 is used for accessing the local and peer knowledgebase. It provides a search field for inputs and checkboxes to select peer or local search. The result page shows search results as a headline summary in their respected classes (e.g. concept, document, project, event, persons, etc.) and as detailed list with key word summary. For each item the "browse" and "link" buttons provide additional information, like members or manager of a project, in a popup box.

11.3. SBS Mid–level Ontology

SBS domain

As introduced in Section 5.4.13, the PIMO represents the organisational setting in the dimensions of people, organizations (customers), topics, documents, and processes. Transferred to the SBS example, an adapted mid-level ontology for the domain of SBS was created. The role of the user (intention) and process (task/situation) define a knowledge space. It includes the customer

(and metadata about him, such as "industrial company"), workflow (e.g. templates) and document (e.g. title, creator, and publisher) agreed organisational knowledge domains (e.g. document type or portfolio) and personal information structures (folder structure e.g. customer-, region-, event-based).

Together with the organizational structures, the PIMO combines both folder structures (Figure 11.1) and represents the users' role and expert level as a model of the users'subjective perspective. The categories expressed in folder structures were represented as PIMO Things using the Protégé tool[3] and added to the ontologies using the Pimo Service (Section 6.2.3).

Role of the user

One user might have different *roles*, which end up in a mixture in his personal workspace structure. To avoid this and not to divert the results of this work, it is assumed that one user has one role and his personal workspace represents the perspective and intention of this role.

However, the model we use (PIMO) is not restricted to representing a single role of a person. Instead, using a context-aware approach that activates a certain role of the user when the user is acting in this role is in principle possible. This is based on the subjective view on above ontologies from the organizational setting: the PIMO is a personal view on these. It is a mediator between the mental model of the user, and the documents of the company. The current role of the user is modelled using the PIMO and RDF, and then the relations of documents and projects to this role can be captured using RDF links. For our experiment, only a very limited PIMO was constructed, it is a step towards representing the user's mental model.

11.4. Procedure

For the test, SBS-internal non-restricted data (102 files) downloaded from the LiveLink knowledgebase is used completed with 23 anonymised SBS management summaries (representing proposal stories) and 30 manually created files (project plans, calculations, contacts and references). These parts represent the basic set of documents. The file structures are part of the user's subjective view derived from directory, email, bookmark etc. structures. The individual file structures of the respective roles with the given basic set of documents are filled according to different scenarios.

Performing searches with different typical queries from SBS operations like "helpdesk", "call center", "customer centralization", etc., "cost reduction" is used as evaluation procedure. For each search result, precision and recall values were measured based on a manual "gold standard" annotation done by the knowledge workers [4]. The experiment is repeated for each scenario and compared to existing search engines. When returning results, the search engine determines the length of the result set by not returning results below a quality threshold. The varying result set length influences the sum of precision and recall and can theoretically result in both recall and

[3] `http://protege.stanford.edu/`

[4] According to Brünken [Bru98] the following sets are the base for measures in recall and precision: M is the set of all relevant objects as part of all system objects, P is the set of retrieved documents and objects (search result), Ma is the set of retrieved relevant documents and objects (relevant search results).

precision being 1.0.

11.5. Results for Each Scenario

Different forms of collaboration exist between sales and proposal managers depending on the combination of peers (different roles and perspectives) as well as availability and similarity of objects (different peers and knowledge objects). They help to outline four scenarios, their differences are in role and perspective will have a specific influence on the search quality and information relevance.

- The first scenario is **S1–Local search**. In direct comparison to Google Desktop one role (e.g. sales manager) searches his own desktop and the organizational database, now applying his native structures within the PIMO.
- The second scenario is **S2–Group search**, searching within data of similar roles working with mostly different topics and files (e.g. customers), within the same domain (e.g. calculation, trends, solution design, etc.).
- The third scenario is **S3–Closed community**, searching within different roles working with similar topics and files (e.g. around business development within a customer community). Especially customer communities are weak structured working frames. They tie together people with different roles (sales manager, solution designer, project manager, etc.), all dealing with the same customer.
- The last scenario is **S4–Open community**, searching within different roles working with different topics and files (e.g. Internet, Intranet). Information is shared from an organizational point of view.

The detailed results are published in [SSSD06] and are included in Mark Siebert's dissertation[5]. A comparison of search results in S2 and S3 is shown in Figure 11.2. For S1, both precision and recall values are high for the individual evaluated keywords. Compared to LiveLink search, especially recall increases using the Brainfiler engine, and additionally increases when using the rule-based search engine. In comparison to S1, precision stays high in S2, but recall drops. For S3, recall stays high but precision drops. In S4, both values drop, compared to S1. The results show that the Semantic Desktop can improve precision by considering the role of an individual workspace (S1 and S2). Deeper integration within a Social Semantic Desktop could expand these to an even larger variety of roles (S3 and S4). Precision depends on the degree of publicity of the searched roles within the local environment. Recall depends on the amount of objects within the knowledge base and increases less with the increased size of the knowledge base.

The use of the *group PIMO* allows to share individual roles and domains amongst peers, improving search quality compared to the existing LiveLink system used at SBS. Using the

[5]Mark Siebert is working on his dissertation at the time of writing.

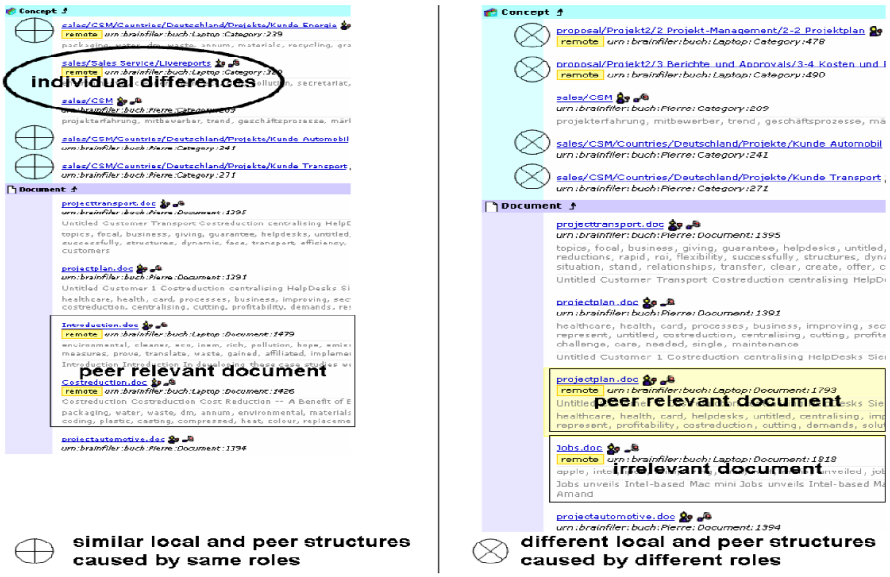


Figure 11.2.: Result visualization of peer search S2 (left) and S3 (right)

formal PIMO model, RDF as data format and a combination of SPARQL and forward chaining rules allows to formalize knowledge about the search process for individual roles.

11.6. Discussion

The tests and published results [SSSD06] proof the positive influence of the Personal Information Model, representing roles and perspectives, on search quality. Qualitative analysis furthermore emphasizes the impact of complete publicity of the roles information model on search quality. Semantic Desktop thus avoids additional editorial or communication efforts, which are required today without semantic search in proposal development.

The PIMO in combination with the Search Service are a good base to integrate personal views into process-related, task- and competence-oriented role concepts from an informational point of view respecting the subjective character of knowledge. Future research will have to provide solutions for enabling second and third level of abstraction, like mental models and mental imagery. This might add a piece to the shift from a reactive to an active search support and information provision, e.g. within the CRM Sales assistant from SAP (“related documents”). Displaying search hits in a process-dependent template structure of a management summary (market trends, business scenarios and customer requirements, compelling events, cost driver, solution, benefits) provides a pre-structured base for further content retrieval from the found documents. Based on the meta-data “document type”, e.g. market information, could be shown in the category “market trends”.

In total, personal information models (personalization, individual roles), semantic search (combining fulltext search, similarity engine, and semantic search), and Semantic Desktop ser-

vices (mash-ups, user observation, peer-to-peer, content management, semantic workflow systems) are the three elements driving further research in this area leading to better search results and reduced response times and lower process costs.

IV

Conclusions

CHAPTER 12
Conclusions

Approach

In this thesis I have proposed a *software architecture model* for the Semantic Desktop consisting of *ontologies*, *services*, and *applications*. Compared to other related studies, this approach has a strong focus on applicability in real world scenarios. Instead of replacing existing applications, they are augmented with semantic features.

12.1. Key results

The core of this work is an integrated representation of all data necessary for PIM, within the *Personal Information Model* (PIMO). A set of services implements functionalities common for PIM on the Semantic Desktop. A unified approach to PIM can now be embedded in various existing applications using plugins, or completely new applications can be build. The approach was evaluated in end-user experiments in order to find out how PIMO reflects the personal mental model of the user. It was verified to support the users in structuring their documents across applications according to their mental model and in retrieving information based on these structures.

Free software

As part of our approach to software engineering at DFKI, we released our source code as free software. Other researchers have used our prototype as a basis for their research and have provided us with valuable feedback. We initiated the integrated EU research project NEPOMUK. Within this project the presented results were integrated into the popular Linux desktop KDE (version 4) and are currently shipped to millions of users. The Aperture framework, initiated as part of this thesis, was downloaded more than 10.000 times. Our prototypes have shown their benefits in many person-years of productive usage, details about these benefits were presented in the case studies.

12.1.1. The Personal Information Model

PIMO

The core is the PIMO approach to the integration of data. Consisting of the parts—*Basic, Upper, and Mid*—it provides constructs to represent mental concepts as *Things* in an ontology. The *basic* part defines that things are labelled and identified in order to represent the personal view of the user, and then can be grounded in existing information elements (such as files on the harddisk). The *upper* ontology defines generic concepts for information found in PIM. Classes independent of application scenario, culture, or domain are defined. Mid-level ontologies are refinements of the upper ontology for known domains. The model is a combination of best practices from RDF, SKOS, and XML Topic Maps.

12.1.2. A Software Architecture for the Semantic Desktop

Services

The proposed software architecture consists of multiple services on which applications are built. Services are defined to store information (*RDF store*), to integrate existing data into a coherent model (*data wrapper*), to build and extend the model (*PIMO service*), and to search for data combining fulltext search and metadata search (*search service*). Based on these services and training data, the *categorization service* can give suggestions how to annotate and file new information. A simplified access for developers is given in the *tagging service*. With the *personal wiki service*, semantic notes can be written independent of application. The *user work context service* gathers information about what the user is currently doing. Finally, *user interface services* make it possible to trigger applications when needed. All these services work on the same data model, PIMO. The clear separation of services from user interface is not found in related Semantic Desktop architectures. Other approaches such as *OpenIris* and *Haystack*[QHK03, CPG05] bundle services and user interface as one and enforce the user to do their work within these new applications.

Personalization

The personalized training of concepts using the categorization service throughout all applications distinguishes the approach.

12.1.3. Evaluation of User Interfaces

Based on early evaluations, we identified the "auxiliary posture" metaphor as a useful approach for a minimalistic user interface. Related work ([QHK03, CPG05]) is based on a "sovereign posture".

Usability Evaluation

At DFKI, we evaluated our assertion that the gnowsis Semantic Desktop will improve PIM of knowledge workers. In a two-month experiment with eight participants, our prototype was used in daily practice. It supports the user in both structuring their documents by providing a semantic search and the possibility for cross-application linking of resources. The integrated semantic wiki was able to fill the authoring gap and provides a possibility for entering unstructured as well as structured text. The PIMO ontology provides a way for users to express their mental model.

12.1.4. Evaluation of long-term use

One question was, how the Gnowsis approach would influence long-term PIM behaviour in the fields of filing, re-finding, and thinking.

The contextual interviews with two long-term users (22 months of usage) brought up the key effects of the software. The *Miniquire* view was the main entry-point to data, as expected and planned by us beforehand. The *personal semantic wiki* proved to be a key user interaction metaphor. Again we had expected this and were more than surprised about the creative ways the users found to use the wiki. For the PIMO ontology, the modelling approach proved to be supportive for PIM behaviour and users have developed ways of orienteering in their data. This again did not surprise us but rather proved our own expectations, and the research of others in the fields of PIM. The great value of *simple relations* to re-find information is a key result of this study, confirming a wise statement that is often heard in Semantic Web research: "*a little semantics goes a long way*".

12.1.5. Evaluation of Benefit and Need

Given the results of this thesis, how does it match the needs of today's knowledge workers and where is the key benefit? Interviews were done with 22 participants from the Kaiserslautern area to identify potential areas where the Semantic Desktop can be useful. Quantitative results were analysed to define and visualize the areas where uses manage important data and are not satisfied with their current system. After a demo of the Semantic Desktop, uses were again interviewed how the software would help them. Qualitative data was gathered as verbal feedback. For the area of e-mail management, participants who saw a benefit of the Semantic Desktop in this area also saw a benefit in a unified folder structure across application borders. The possibility to annotate and relate persons was often mentioned in the verbal answers as the main added value of the Semantic Desktop. Together, the results of this evaluation showed where and how the Semantic Desktop can bring an immediate benefit.

12.1.6. Evaluation in Proposal Management

Another evaluation of the system was conducted for a special application domain, proposal management at Siemens Business Services.

Increasing Search Quality

The PIMO in combination with the *search service* were used to annotate the knowledge based on the roles of the users (proposal manager and sales manager). By engaging different rules in the retrieval engine for each role, the search engine was further personalized. Precision and recall measures of search results improved compared to the already deployed system. The approach of Semantic Desktop services proved to be applicable in the experimental setup.

12.1.7. Evaluation Critique

Looking at the *limited* results that were achieved by the questionnaire and the evaluation logging, and the *rich* information about practical experiences retrieved by video-recording contextual interviews, more evaluations with interviews could be done.

Both *methods* were used for evaluation. The long term case study and the contextual inquiry brought insight about how users cope with Semantic Technology. Besides that, the technical effort is not to be underestimated. For example, participants had the possibility to integrate their MS-Outlook e-mails, contacts, and appointments with gnowsis, but didn't use this option because installation was too complicated and they feared that bugs could damage their data. Hence, software problems will always influence your evaluation.

12.1.8. Community Involvement

NEPOMUK

The proposed architecture served as input for the NEPOMUK project [GHM+07]. PIMO, the services, and applications are changed and reworked thoroughly in this project. Other researchers have used our software architecture and prototype implementation as a basis for their work.

SQAPS

Norberto Fernandez started with the assumptions of the PIMO and created *SQA4Desktop*

which tries to utilize the information that the user provides while performing Web searches [FGSSB06]. Basically this system requires the user to annotate his or her query by associating a concept or set of concepts to it. The concepts involved in this process need to be taken from Wikipedia. The concepts from Wikipedia are again represented as things in the user's PIMO, and can be used to annotate resources found in the web search. Apart from populating the PIMO, the approach is useful in resource annotation.

SeMouse In the SeMouse project, existing desktop applications are augmented to potential ontology editors [IAD06]. The mouse is turned into a semantic device, which is used to connect text editing software with an ontology. This approach was successfully ported to work with PIMO and the gnowsis architecture. Vinh Tuan Thai et al. have developed a graphical document browser to explore large document spaces based on the PIMO ontology and published their work at

On Mobile ESWC 2006 [THD08]. Woerndl and Woerl ported the PIMO idea to mobile devices, studying how ubiquitous access to personal structures can support mobile users [WW08].

Workshops A series of workshops[1] helps to further enlarge the community around the Semantic Desktop.

12.2. Comparison with Related Work

Problem Current desktop operating systems do not provide the means to express knowledge independent of application or domain. File-systems, e-mail folders, or tags each replicate parts of the mental model of the user, but often in parallel and unbeknown to each other. The user has to express the same ideas repeatedly, for each application. The problem of PIM increases with the amount of digital information gathered on personal computers. In theory, users could carry all their digital information on a portable computer with them, in practice it is a problem to find and manage information within such a collection.

Related Work These problems are addressed in various scientific fields: Semantic Web, Semantic Desktop, cognitive science, HCI, and PIM. In PIM, the work by Boardman shows that integration across applications is possible and feasible [Boa04]. Previous work by the author [Sau03, SGK+06, SvED07] is placed in the research field of the *Semantic Desktop* [DF04, SBD05]. In this field, related projects have covered various aspects *e. g.*, *Haystack* provides an integrated platform for PIM [QHK03], *CALO* and *OpenIris* the use of machine learning and artificial intelligence [CPG05], and *Semex* (SEMantic EXplorer) [DH05] capable algorithms for data analysis and reference reconciliation.

Effect on PIM Tools

Looking at the various requirements of PIM research stated in Section 6.1, we can see that many of these requirements are now realized or at least simplified when applications are built upon Semantic Desktop services and the PIMO ontology. Let us revisit the definition of PIM given in Section 2.2, which states *PIM as the management of data in the personal knowledge space as performed by the owning individual*.

[1] A list is given in the introduction in Section 1.1

Unified Information Space

Through the data wrapper services, it is possible to represent and integrate information from the personal knowledge workspace. Based on the Semantic Web standards (URI, RDF, ontologies) and PIMO, this data can now be annotated in a uniform way, and the annotations stored in the RDF store service. This allows PIM tools to apply their functionality on a unified information space, a requirement strongly requested by the PIM community.

Filing and Finding

Filing is supported by the *tagging service* and *categorization service*, allowing to use the same filing approach and categories independent of application. An example application, the *drop-box* shows these features. Finding information is supported by the *search service* and new ways of browsing applications are possible using the semantic relations of search results. The existing finding approaches of structural browsing and fulltext search are combined.

It is now possible to implement the state of the art in PIM using our proposed architecture, combining them. To nurture communication between the PIM and Semantic Desktop communities, I participated in the PIM Workshop at CHI 2008 conference.

12.3. Outlook and Open Questions

The architecture presented in this thesis is the result of many experiments, more will follow by ourselves and other researchers, invalidating our results or refining them with new insights. Many of the posed open questions are addressed within the NEPOMUK project.

Some of the presented ontologies, services and applications are deployed as part of the release 4.0 of the NEPOMUK-KDE desktop for the Linux operating system. This implementation is managed by Sebastian Trüg. It allows users to benefit from the presented ideas, and developers to create more interesting applications. With the deployment of KDE 4.0, the Semantic Desktop and the idea of the PIMO will be delivered to more than a million users, which is partly a success of our work and the Semantic Web, but also opens a challenging field for research.

12.3.1. Connecting the Semantic Desktop with the Semantic Web 2.0

There is much research on the subject of "web 2.0", the "social web", "web 3.0" and "Semantic Web", for short "Semantic Web 2.0". There are several promising conjunctions:

- As the Semantic Desktop now serves as a programming platform similar to that of existing web 2.0 services, a new kind of application is possible, the *desktop mash-up*. Small, agile applications that build their power based on combining desktop applications through the semantic interfaces are now feasible. Such applications may change business models, distribution channels, social behaviour, or inflict new security and privacy issues.

- Algorithms developed for the Semantic Web 2.0 can be ported to the Semantic Desktop. For example page-rank can be ported to the desktop, by using other edges besides hyperlinks; also co-occurrence analysis of tags for tag suggestions is now possible.

- And most important, the data, services, and applications offered by Semantic Web 2.0 applications must be integrated with data on the Semantic Desktop. We did not define a service for integration of the PIMO with concepts defined in applications running on the web, this is a subject for future work.

12.3.2. Algorithms creating PIMO Structures

reference reconciliation

In this thesis, no algorithms were presented that automatically generate PIMO structures based on data extracted by the data wrapper service. Experiments have been done within the Knowledge Management group at DFKI to analyse the semantic meaning of file-system structures and generate PIMO structures from those. Others are also working on this problem. In the work of Xin Dong et al. [DH05] we find a good entry point on *reference reconciliation* which needs to be continued. Ioannou et al. created and evaluated statistical methods for *entity linkage* on the Semantic Desktop in [INN08].

NLP

Also, *natural language processing* tools can be used to generate PIMO structures. The knowledge stored in free text is a key to a large-scale adoption of the Semantic Desktop. Users already have a collection of text documents and e-mails which can be analysed.

12.3.3. Improved User Interaction and New Information Metaphors

HCI

In the field of Semantic Web, various promising approaches exist for information visualization, and interactive exploration, such as *faceted browsing*(a nice example of a faceted browser being [HvOH06]). These can now also be used on the desktop, as the interfaces and data structures are the same.

3d interaction

In the author's personal view, the three-dimensional interaction metaphor should be further explored to navigate the personal mental model. Evaluations have shown that productivity does not rise significantly when user interfaces use three dimensions, while the interaction metaphor and representation metaphor remain the same (windows and buttons). Given the possibility of the Semantic Web assigning a three-dimensional position to any information element (as any other annotation in RDF), the problem of an *architecture of information spaces* arises. Today, three-dimensional games and platforms attract more attention and a bigger audience than ever before [2]. How can personal information be visualized and made available in collaborative, three dimensional environments, and what implications to knowledge work does this have?

[2] The artificial world "Azeroth" of World Of Warcraft has nine million human inhabitants in August 2007, more than Austria. In Second Life, about one million US-dollar is transferred between users each day, a total of 14 million dollars is bound inside this economy measured on one day in December 2007.

12.4. Final Remarks

This work is, I hope, a useful contribution to the field of Semantic Web and desktop computing. The accompanying implementations show that the approach is implementable and provides essential functionality for the desktop. For the future, other researchers and engineers can extend and build upon our work, creating software that supports people in *managing their personal information* the way it was envisioned long ago by Vannevar Bush and dreamt by Neal Stephenson and Vernor Vinge.

V

Appendices

APPENDIX A
Appendix

A.1. Validation Rules of PIMO

```
#-----------------------------------------------------------------
# PIMO Checking Rules - is a PIMO valid?
#-----------------------------------------------------------------
# for a good documentation of this, see:
# http://jena.sourceforge.net/inference/index.html#RULEhybrid
#
# for other validation checks see the Jena rules in
# jena.jar/etc/*

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix pimo_api:
  <http://ontologies.opendfki.de/repos/ontologies/pim/pimo-api#>.
@prefix pimo:
  <http://ontologies.opendfki.de/repos/ontologies/pim/pimo#>.
@prefix protege: <http://protege.stanford.edu/system#>.
@prefix test:
  <http://ontologies.opendfki.de/repos/ontologies/pim/TESTONTOLOGY#>.


#-----------------------------------------------------------------
# CHECKING
#-----------------------------------------------------------------

[allRelatingPropertiesNeedInverse:
(?x rb:violation error('inverse property check',
  'All relating properties need an inverse', ?x))
<- (?x rdfs:subPropertyOf pimo:relatingProperty),
 noValue(?x protege:inverseProperty ?y)]


# Check domain and range
(?x rdfs:subPropertyOf pimo:relatingProperty) ->
```

```
(?x test:checkDomainAndRange 1).
-> (rdf:type test:checkDomainAndRange 1).
-> (pimo:occurrence test:checkDomainAndRange 1).

[checkDomain:
(?x rb:violation error('domain',
 'The used domain does not fit, see triple.', ?p, ?y, ?c))
<- (?p test:checkDomainAndRange 1), (?x ?p ?y),
(?p rdfs:domain ?c),
(?x rdf:type ?tt), noValue(?tt rdfs:subClassOf ?c)]

[checkRange:
(?x rb:violation error('range',
'The used range does not fit, see triple.', ?x, ?p, ?y))
<- (?p test:checkDomainAndRange 1), (?x ?p ?y),
(?p rdfs:range ?c),
(?y rdf:type ?tt), noValue(?tt rdfs:subClassOf ?c)]


# check protege-cardinality
[checkRequiredProperties:
(?x rb:violation error('required missing',
 'The object o of type t is missing the required property p.
  (o, p, t).',
 ?x, ?p, ?t))
<- (?x rdf:type ?t), (?t rdfs:subClassOf ?st),
   (?p rdfs:domain ?st),
 (?p protege:minCardinality '1'),
noValue(?x ?p ?y), noValue(?p protege:defaultValues ?_d) ]

# check labels
[checkLabelThings:
(?x rb:violation error('label', 'Things need a rdfs:label', ?x))
<- (?x rdf:type pimo:Thing),
 (?p rdfs:range ?c), noValue(?x rdfs:label ?y)]

[checkLabelClass:
(?x rb:violation error('label',
  'PimoClasses need a rdfs:label', ?x))
<- (?x rdf:type pimo:PimoClass),
 (?p rdfs:range ?c), noValue(?x rdfs:label ?y)]
```

```
#-----------------------------------------------------------------
[checkEveryThingHasAType1:
(?x rb:violation error('no type(o)',
 'everything need a type (object use)', ?x))
<- (?s ?p ?x), notLiteral(?x),
 noValue(?p rdfs:subPropertyOf pimo:describingProperty),
 noValue(?x rdf:type ?y)]

[checkEveryRelationInverse:
(?x rb:violation error('no inverse relation',
 'every relation needs its inverse', ?x, ?p, ?y))
<- (?p protege:inverseProperty ?i), (?x ?p ?y),
  noValue(?y ?i ?x)]
```

A.2. Paul's PIMO

Here you find the RDF representation of the example scenario, *Paul's PIMO*. It was used as illustrative example in some publications, and also as input for automatic testing of the software.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf
'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY pimo
'http://ontologies.opendfki.de/repos/ontologies/pim/pimo#'>
<!ENTITY a
'http://protege.stanford.edu/system#'>
<!ENTITY pimo_
'gnowsis://paul@example.com/resources/pimo/'>
<!ENTITY rdfs
'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
 xmlns:pimo="&pimo;"
 xmlns:a="&a;"
 xmlns:pimo_="&pimo_;"
 xmlns:rdfs="&rdfs;">
<pimo:PimoClass rdf:about="&pimo_;BusinessPlan"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="BusinessPlan">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<rdfs:subClassOf rdf:resource="&pimo;Document"/>
```

```
</pimo:PimoClass>
<pimo:PimoClass rdf:about="&pimo_;Flyer"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="Flyer">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<rdfs:subClassOf rdf:resource="&pimo;Document"/>
</pimo:PimoClass>
<pimo:PimoSlot rdf:about="&pimo_;isManagedBy"
 pimo:metaViewLevel="SYSTEM-USER"
 a:maxCardinality="1"
 rdfs:label="isManagedBy">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<a:inverseProperty rdf:resource="&pimo_;manager"/>
<rdfs:subPropertyOf rdf:resource="&pimo;partOf"/>
<rdfs:range rdf:resource="&rdfs;Resource"/>
</pimo:PimoSlot>
<pimo:PimoSlot rdf:about="&pimo_;isSupervisedBy"
 pimo:metaViewLevel="SYSTEM-USER"
 a:maxCardinality="1"
 rdfs:label="isSupervisedBy">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<a:inverseProperty rdf:resource="&pimo_;supervisor"/>
<rdfs:subPropertyOf rdf:resource="&pimo;partOf"/>
<rdfs:range rdf:resource="&rdfs;Resource"/>
</pimo:PimoSlot>
<pimo:PimoSlot rdf:about="&pimo_;manager"
 pimo:metaViewLevel="SYSTEM-USER"
 a:maxCardinality="1"
 rdfs:label="manager">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<a:inverseProperty rdf:resource="&pimo_;isManagedBy"/>
<rdfs:range rdf:resource="&pimo;Person"/>
<rdfs:subPropertyOf rdf:resource="&pimo;hasPart"/>
</pimo:PimoSlot>
<pimo:PimoSlot rdf:about="&pimo_;participant"
 pimo:metaViewLevel="SYSTEM-USER"
 a:maxCardinality="1"
 rdfs:label="participant">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<a:inverseProperty rdf:resource="&pimo_;participatesAt"/>
<rdfs:range rdf:resource="&pimo;Person"/>
<rdfs:subPropertyOf rdf:resource="&pimo;hasPart"/>
```

```
</pimo:PimoSlot>
<pimo:PimoSlot rdf:about="&pimo_;participatesAt"
 pimo:metaViewLevel="SYSTEM-USER"
 a:maxCardinality="1"
 rdfs:label="participatesAt">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<a:inverseProperty rdf:resource="&pimo_;participant"/>
<rdfs:subPropertyOf rdf:resource="&pimo;partOf"/>
<rdfs:range rdf:resource="&rdfs;Resource"/>
</pimo:PimoSlot>
<pimo:PimoSlot rdf:about="&pimo_;supervisor"
 pimo:metaViewLevel="SYSTEM-USER"
 a:maxCardinality="1"
 rdfs:label="supervisor">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<a:inverseProperty rdf:resource="&pimo_;isSupervisedBy"/>
<rdfs:range rdf:resource="&pimo;Person"/>
<rdfs:subPropertyOf rdf:resource="&pimo;hasPart"/>
</pimo:PimoSlot>
<rdf:Property rdf:about="&pimo;isGroundingOccurrenceOf"
 rdfs:label="pimo:isGroundingOccurrenceOf"/>
<rdf:Property rdf:about="&pimo;isOccurrenceOf"
 rdfs:label="pimo:isOccurrenceOf"/>
</rdf:RDF>
```

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf
 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY pimo
 'http://ontologies.opendfki.de/repos/ontologies/pim/pimo#'>
<!ENTITY photos
 'file:///C:/myDocuments/Paul/documents/projects/branchinrome/photos/'>
<!ENTITY pimo_
 'gnowsis://paul@example.com/resources/pimo/'>
<!ENTITY branchinrome
 'file:///C:/Documents/Paul/documents/projects/branchinrome/'>
<!ENTITY babelfish_altav
 'http://babelfish.altavista.com/'>
<!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
 xmlns:pimo="&pimo;"
 xmlns:photos="&photos;"
 xmlns:pimo_="&pimo_;"
 xmlns:branchinrome="&branchinrome;"
 xmlns:babelfish_altav="&babelfish_altav;"
 xmlns:rdfs="&rdfs;">
<pimo:FileFolder rdf:about="&branchinrome;"
 pimo:label="FilesBranchOffice"
 pimo:uri="file://c:/myDocuments/Paul/documents/projects/branchinrome/"
 rdfs:label="FilesBranchOffice"/>
<pimo:ImageFile rdf:about="&photos;image23.jpg"
 pimo:label="picture23"
 rdfs:label="picture23"/>
<pimo:AddressBookCard rdf:about="&pimo_;PaulsAddress"
 pimo:label="PaulsAddress"
 rdfs:label="PaulsAddress"/>
<pimo:PersonalInformationModel rdf:about="&pimo_;PaulsPim"
 pimo:isWriteable="false"
 rdfs:label="PaulsPim">
<pimo:metaDefines rdf:resource="&pimo_;BusinessPlan"/>
<pimo:metaDefines rdf:resource="&pimo_;Flyer"/>
<pimo:hasNamespace rdf:resource="&pimo_;PaulsPim-Namespace"/>
<pimo:metaDefines rdf:resource="&pimo_;isManagedBy"/>
<pimo:metaDefines rdf:resource="&pimo_;isSupervisedBy"/>
<pimo:metaDefines rdf:resource="&pimo_;manager"/>
```

```
<pimo:metaDefines rdf:resource="&pimo_;participant"/>
<pimo:metaDefines rdf:resource="&pimo_;participatesAt"/>
<pimo:metaDefines rdf:resource="&pimo_;instance10000"/>
<pimo:metaDefines rdf:resource="&pimo_;instance14"/>
<pimo:metaDefines rdf:resource="&pimo_;instance20"/>
<pimo:metaDefines rdf:resource="&pimo_;instance21"/>
<pimo:metaDefines rdf:resource="&pimo_;instance22"/>
<pimo:metaDefines rdf:resource="&pimo_;instance23"/>
<pimo:metaDefines rdf:resource="&pimo_;instance24"/>
<pimo:metaDefines rdf:resource="&pimo_;instance25"/>
<pimo:metaOwner rdf:resource="&pimo_;instance25"/>
<pimo:metaDefines rdf:resource="&pimo_;instance26"/>
<pimo:metaDefines rdf:resource="&pimo_;instance27"/>
<pimo:metaDefines rdf:resource="&pimo_;instance28"/>
<pimo:metaDefines rdf:resource="&pimo_;instance29"/>
<pimo:metaDefines rdf:resource="&pimo_;instance30"/>
<pimo:metaDefines rdf:resource="&pimo_;instance5"/>
<pimo:metaDefines rdf:resource="&pimo_;supervisor"/>
<pimo:metaImports rdf:resource="&pimo;PIMO-language"/>
</pimo:PersonalInformationModel>
<pimo:Namespace rdf:about="&pimo_;PaulsPim-Namespace"
 pimo:namespaceAbbreviation="paul"
 pimo:namespaceUri="gnowsis://paul@example.com/resources/pimo/"
 rdfs:label="PaulsPim-Namespace"/>
<pimo:File rdf:about="&pimo_;instance0"
 pimo:hasMimeType="application/ms-word"
 pimo:label="Business Plan Document"
 rdfs:label="Business Plan Document">
<pimo:uri>
file:///C:/myDocuments/Paul/documents/businessplans/romeplan.doc
</pimo:uri>
</pimo:File>
<pimo:PimoWikiContent rdf:about="&pimo_;instance10000"
 pimo:wikiLastModified="20060202T13:54:00"
 pimo:wikiText="Start a new branch office in Rome."
 pimo:wikiVersionNumber="1"
 rdfs:label="instance10000"/>
<pimo:PeriodOfTime rdf:about="&pimo_;instance0"
 pimo:beginTime="20060101T08:00"
 pimo:endTime="20060101T18:00"
 rdfs:label="instance0"/>
<pimo:Email rdf:about="&pimo_;instance10"
```

```
 pimo:label="Project Status"
 rdfs:label="Project Status"/>
<pimo:Topic rdf:about="&pimo_;instance10000"
 pimo:label="Marketing"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="Marketing">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
</pimo:Topic>
<pimo:Email rdf:about="&pimo_;instance11"
 pimo:label="email4"
 rdfs:label="email4"/>
<pimo_:Flyer rdf:about="&pimo_;instance14"
 pimo:label="InvitationFlyerRome"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="InvitationFlyerRome">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:hasTopic rdf:resource="&pimo_;instance24"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
</pimo_:Flyer>
<pimo:Webpage rdf:about="&pimo_;instance17"
 pimo:label="Comune di Roma | Sito Istituzionale"
 pimo:uri="http://www.comune.roma.it/"
 rdfs:label="Comune di Roma | Sito Istituzionale"/>
<pimo:EMailFolder rdf:about="&pimo_;instance19"
 pimo:label="EmailsBranchOffice"
 pimo:uri="imap://paul@example.com/Projects/BranchOffice/"
 rdfs:label="EmailsBranchOffice"/>
<pimo:AddressBookCard rdf:about="&pimo_;instance2"
 pimo:label="PetersAddress"
 rdfs:label="PetersAddress"/>
<pimo:Building rdf:about="&pimo_;instance20"
 pimo:label="OfficeBuildingInRome"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="OfficeBuildingInRome">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
</pimo:Building>
<pimo:City rdf:about="&pimo_;instance21"
 pimo:label="Athens"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="Athens">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
</pimo:City>
```

```
<pimo:City rdf:about="&pimo_;instance22"
 pimo:label="Rome"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="Rome">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:related rdf:resource="&pimo_;instance24"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
<pimo:occurrence rdf:resource="&babelfish_altav;tr"/>
</pimo:City>
<pimo:Organization rdf:about="&pimo_;instance23"
 pimo:label="AcmeCorp"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="AcmeCorp">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
</pimo:Organization>
<pimo:Project rdf:about="&pimo_;instance24"
 pimo:label="BranchOfficeRome"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="BranchOfficeRome">
<pimo:hasContainer rdf:resource="&branchinrome;"/>
<pimo:occurrence rdf:resource="&photos;image23.jpg"/>
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:hasWikiContent rdf:resource="&pimo_;instance10000"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance14"/>
<pimo:isTopicOf rdf:resource="&pimo_;instance14"/>
<pimo:hasContainer rdf:resource="&pimo_;instance19"/>
<pimo:related rdf:resource="&pimo_;instance22"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance22"/>
<pimo:hasPart rdf:resource="&pimo_;instance25"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance25"/>
<pimo:hasPart rdf:resource="&pimo_;instance26"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance26"/>
<pimo:hasPart rdf:resource="&pimo_;instance27"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance27"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance29"/>
<pimo:isTopicOf rdf:resource="&pimo_;instance29"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance30"/>
<pimo:isTopicOf rdf:resource="&pimo_;instance30"/>
</pimo:Project>
<pimo:Person rdf:about="&pimo_;instance25"
 pimo:label="Paul"
 pimo:metaViewLevel="SYSTEM-USER"
```

```
 rdfs:label="Paul">
<pimo:occurrence rdf:resource="&pimo_;PaulsAddress"/>
<pimo:metaCreationSupportedBy rdf:resource="&pimo_;PaulsAddress"/>
<pimo:metaOwnsPimo rdf:resource="&pimo_;PaulsPim"/>
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:partOf rdf:resource="&pimo_;instance24"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
<pimo:related rdf:resource="&pimo_;instance30"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance30"/>
<pimo:attends rdf:resource="&pimo_;instance30"/>
</pimo:Person>
<pimo:Person rdf:about="&pimo_;instance26"
 pimo:label="Peter"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="Peter">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:occurrence rdf:resource="&pimo_;instance2"/>
<pimo:metaCreationSupportedBy rdf:resource="&pimo_;instance2"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
<pimo:partOf rdf:resource="&pimo_;instance24"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance30"/>
<pimo:attends rdf:resource="&pimo_;instance30"/>
<pimo:related rdf:resource="&pimo_;instance30"/>
</pimo:Person>
<pimo:Person rdf:about="&pimo_;instance27"
 pimo:label="Tim"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="Tim">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
<pimo:partOf rdf:resource="&pimo_;instance24"/>
<pimo:metaCreationSupportedBy rdf:resource="&pimo_;instance3"/>
<pimo:occurrence rdf:resource="&pimo_;instance3"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance30"/>
<pimo:attends rdf:resource="&pimo_;instance30"/>
<pimo:related rdf:resource="&pimo_;instance30"/>
</pimo:Person>
<pimo:SocialEvent rdf:about="&pimo_;instance28"
 pimo:label="GrandOpeningRomeBranchofAcmeCorp"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="GrandOpeningRomeBranchofAcmeCorp">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
```

```
<pimo:duration rdf:resource="&pimo_;instance0"/>
</pimo:SocialEvent>
<pimo:Meeting rdf:about="&pimo_;instance29"
 pimo:label="KickoffMeetingRome"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="KickoffMeetingRome">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:hasTopic rdf:resource="&pimo_;instance24"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
<pimo:occurrence rdf:resource="&pimo_;instance6"/>
<pimo:metaCreationSupportedBy rdf:resource="&pimo_;instance6"/>
</pimo:Meeting>
<pimo:AddressBookCard rdf:about="&pimo_;instance3"
 pimo:label="TimsAddress"
 rdfs:label="TimsAddress"/>
<pimo:Meeting rdf:about="&pimo_;instance30"
 pimo:label="ProjectStatusMeetingRome"
 pimo:metaViewLevel="SYSTEM-USER"
 rdfs:label="ProjectStatusMeetingRome">
<pimo:occurrence rdf:resource="&photos;image23.jpg"/>
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:occurrence rdf:resource="&pimo_;instance10"/>
<pimo:occurrence rdf:resource="&pimo_;instance11"/>
<pimo:occurrence rdf:resource="&pimo_;instance17"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance24"/>
<pimo:hasTopic rdf:resource="&pimo_;instance24"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance25"/>
<pimo:related rdf:resource="&pimo_;instance25"/>
<pimo:attendee rdf:resource="&pimo_;instance25"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance26"/>
<pimo:related rdf:resource="&pimo_;instance26"/>
<pimo:attendee rdf:resource="&pimo_;instance26"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance27"/>
<pimo:related rdf:resource="&pimo_;instance27"/>
<pimo:attendee rdf:resource="&pimo_;instance27"/>
<pimo:hasPart rdf:resource="&pimo_;instance5"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance5"/>
<pimo:occurrence rdf:resource="&pimo_;instance9"/>
</pimo:Meeting>
<pimo_:BusinessPlan rdf:about="&pimo_;instance5"
 pimo:label="BusinessPlanRomeBranch"
 pimo:metaViewLevel="SYSTEM-USER"
```

```
 rdfs:label="BusinessPlanRomeBranch">
<pimo:metaIsDefinedBy rdf:resource="&pimo_;PaulsPim"/>
<pimo:occurrence rdf:resource="&pimo_;instance0"/>
<pimo:metaCreationSupportedBy rdf:resource="&pimo_;instance0"/>
<pimo:partOf rdf:resource="&pimo_;instance30"/>
<pimo:relatingProperty rdf:resource="&pimo_;instance30"/>
</pimo_:BusinessPlan>
<pimo:CalendarEvent rdf:about="&pimo_;instance6"
 pimo:label="KickoffMeetingOutlook"
 rdfs:label="KickoffMeetingOutlook"/>
<pimo:CalendarEvent rdf:about="&pimo_;instance7"
 pimo:label="StatusMeetingOutlook"
 rdfs:label="StatusMeetingOutlook"/>
<pimo:Email rdf:about="&pimo_;instance8"
 pimo:label="check potential customers"
 rdfs:label="check potential customers"/>
<pimo:Email rdf:about="&pimo_;instance9"
 pimo:label="Project Status"
 rdfs:label="Project Status"/>
<pimo:Webpage rdf:about="&babelfish_altav;tr"
 pimo:label="AltaVista - Babel Fish  Translation"
 pimo:uri="http://babelfish.altavista.com/tr"
 rdfs:label="AltaVista - Babel Fish  Translation"/>
</rdf:RDF>
```

A.3. Usability Test Scenario

Scenario We give you a tool, which brings the ideas of the Semantic Web on your local Desktop: Gnowsis - the Semantic Desktop. Within the next minutes, starting from the installation and configuration up to the creation of the first semantic relation on your Desktop, you will dive into the world of Semantic Desktop computing.
The first use cases serve the purpose of installation, as well as the familiarization with the system. The subsequent use cases will help you to understand the use and capability of such a system.

Installation We already gave you the Gnowsis installation file. Please perform a double-click on it and install your Gnowsis with the help of the emerging installtion wizard.

Setup After you successfully installed the program, the data sources must be configured. Have a look at the configuration window of Gnowsis and add one or more data source(s) of your choice (e.g. the folder c:\Eigene Dateien/Evaluation).

After the data sources are added, please start the indexing in order to make them available in Gnowsis.

Creation of New "Things" The system is now completely installed and configured, but it does not contain any data (except the data from the data sources). It is up to you to change this now. Therefore we want you to create a PIMO structure of your choice, which consists of at least ten new subclasses (of thing) and instances (things). For example managing a project or planning an event will lead to several new classes and instances.

Linking of Resources Now, since the basic structure was successfully created, the new instances (things) need to be related with resources (files and folders from your filesystem). Therefore please take one or more random files (e.g. one of those located in a data source you just connected, c:\Eigene Dateien/Evaluation). Select a suitable predicate for each relation you create.

Semantic Wiki After you linked some resources, try to use the semantic wiki to describe some the things in detail (e.g. "I created this thing because it was a test and ...").

Saving an Email Attachement or Downloaded a File (optional) We sent you an email with an attached pdf-document. You want to store the new file in your filesystem, but you are too lazy to read you through the whole document to get an idea where to file it.
Use the drop box to locate an adequate folder. Additionally use the drop box to add some more tags to the new file to improve a retrieval later on. You did not get our email and file? Then perhaps you want to download a new paper from the internet directly into the drop box.

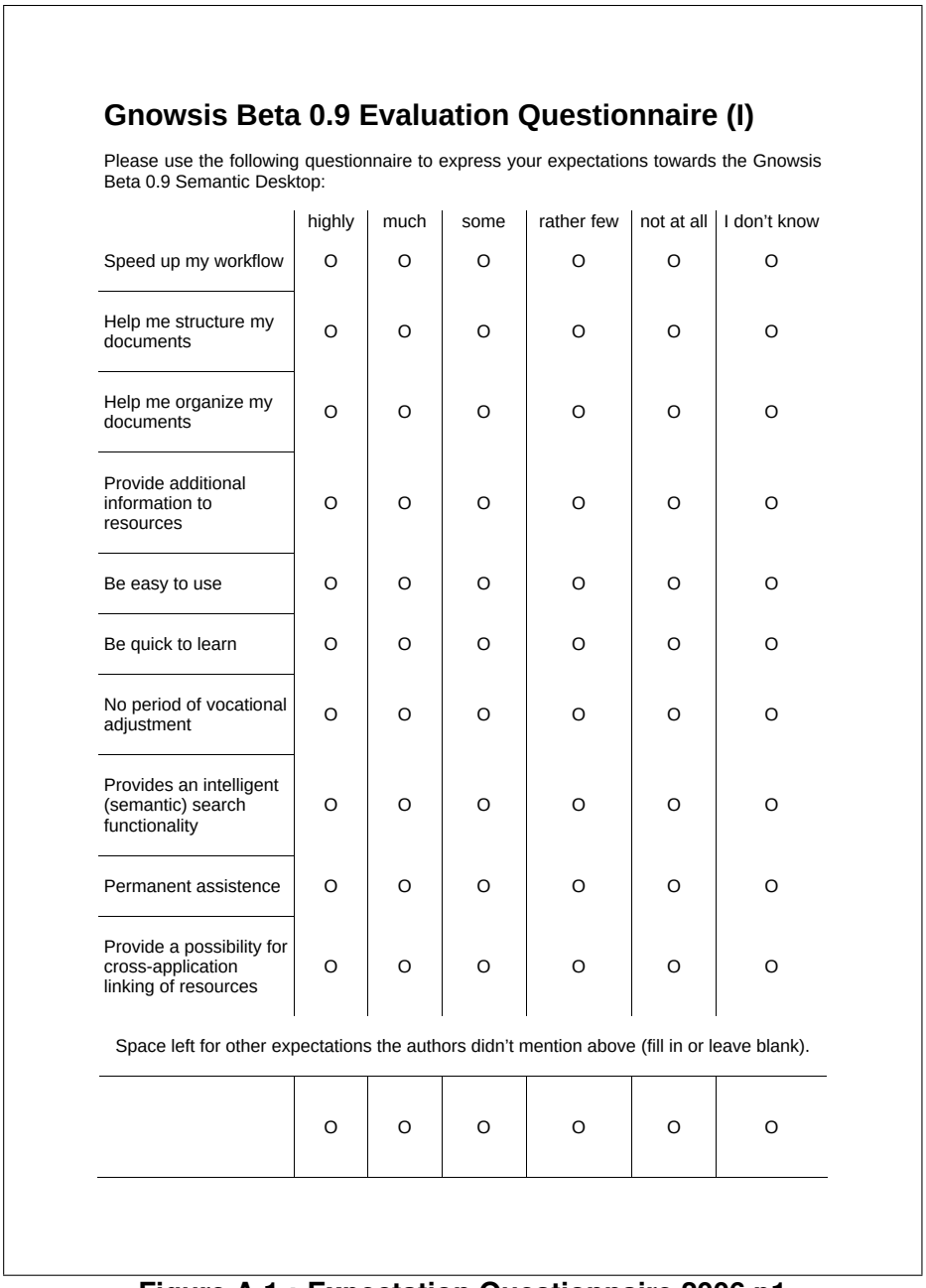
Tagging an Email (optional) Please create an additional instance (thing) named "evaluation". Do you remember the email we send you? Use your Mozilla Thunderbird and our the tagging plugin to relate the email from us with the new created tag "evaluation". Tag at least five other emails.

Search After filling the system with some initial data, you want to have a look at a file (resource) or instance (thing) of your choice. Use the Gnowsis search to retrieve the file (Note, the file has to be either a instance (thing) or a file (resource) within one of the data sources you added. Gnowsis only indexes the data sources listed in the configuration)

The End (of the Usability Test) Thank you for your participation. Now you can go on using the Gnowsis Semantic Desktop for whatever you want. If you have any questions or problems, don't hesitate to ask me or one of the other developers. Have fun using Gnowsis!

A.4. Expectation Questionnaire 2006

This is the expectation questionnaire used in the 2006 Evaluation as described in Section 8.4.



Gnowsis Beta 0.9 Evaluation Questionnaire (I)

Please use the following questionnaire to express your expectations towards the Gnowsis Beta 0.9 Semantic Desktop:

	highly	much	some	rather few	not at all	I don't know
Speed up my workflow	O	O	O	O	O	O
Help me structure my documents	O	O	O	O	O	O
Help me organize my documents	O	O	O	O	O	O
Provide additional information to resources	O	O	O	O	O	O
Be easy to use	O	O	O	O	O	O
Be quick to learn	O	O	O	O	O	O
No period of vocational adjustment	O	O	O	O	O	O
Provides an intelligent (semantic) search functionality	O	O	O	O	O	O
Permanent assistence	O	O	O	O	O	O
Provide a possibility for cross-application linking of resources	O	O	O	O	O	O

Space left for other expectations the authors didn't mention above (fill in or leave blank).

	O	O	O	O	O	O

Figure A.1.: Expectation Questionnaire 2006 p1

	O	O	O	O	O	O
	O	O	O	O	O	O

Now some simple yes/no questions:

01. Do you use a desktop search engine?

O yes, it is called _____

O no

_____

02. Do you contribute to a Wiki regularly?

O yes

O no

_____

03. Do you also use a Wiki to organize yourself?

O yes

O no

_____

04. Do you use tools that help you organize your (local) data (semi-)automatically? (like ???)

O yes, they are called _____

O no

_____

05. Do you use the same structures in different applications (e.g. E-mail application and filesystem)?

O yes

O no

Figure A.2.: Expectation Questionnaire 2006 p2

---

06. Do you use a blog-system to store personal information you won't find again otherwise (like bookmarks, text snppets, etc.)?

O yes

O no

One last question and you're done (for now):

07. What do you think are the most important criterias of a semantic Desktop / semantic Wiki combination? (Check the two things that seem to be most important for you)

[ ] Time saving for the user
[ ] Quality of (user-)support
[ ] Possibility of high level search querries
[ ] Ease of use
[ ] ???

Additional comments / suggestions:

_____

_____

_____

_____

_____

Thank you for filling out the first part of the Gnowsis Beta 0.9 Evaluation questionnaire!

Just watch out for **Dominik Heim** or **Leo Sauermann** and hand him the questionnaire

Figure A.3.: Expectation Questionnaire 2006 p3

A.5. Long-Term Study Interview Guide

In the next pages, the interview guide used for the long-term evaluation is included. The four-page guide was used in the contextual interviews conducted in April 2008, presented in Chapter 9.

Contextual Interview With Gnowsis Long Term Users

Goals: What do we want to find out?

How Gnowsis has affected the way you do Personal Information Management?

How do users file information, search for and retrieve information, and think and organize information?

What implication does this have on efficiency, time, satisfaction, and quality of work?

Observations about your PIMO: what classes did you use, where did you extend PIMO, what did you not use, and why.

The results will be used to improve the software and general observations (anonymized) will be published.

Introduction

The interview is accompanied by a written printout of this document.
We start the Interview by an introduction to the goals. We will start by asking a few general questions about you (the participant), then about your general use of gnowsis. Then I will ask you to do a few tasks using gnowsis and ask you to comment on them.

I will record the interview, and your screen, on video. The videos will be reviewed by me after the interview and not shown to anyone else. I will use statements of you to illustrate the system, anonymized. Is this ok with you?
I want you to use the "think aloud" technique[1]. Important: this is a Software test. You (the user) can do nothing wrong, no matter what you do!

(set up the video-camera, it records the audio of the user and the contents of the screen. Video must capture what they do and show in gnowsis)

Questionnaire

Name:
Gender:
Job Position:
Skill in Programming:
Skill in Semantics:

Since when do you use gnowsis?

How often would you say that you use gnowsis? Daily? Weekly?

1. never
2. once a month
3. once a week
4. multiple days a week
5. often each day

[1] http://en.wikipedia.org/wiki/Think_aloud_protocol

Figure A.4.: Long Term Interview Guide

How many files on your disk are relevant to your Personal Information Model? Could you show them to me and let the Operating System count them?

For what did you use gnowsis last?

PIMO Structures: Can you show us some instances and classes which you have used in the last days?

Why did you create these instances?

Did you associate files?

Web-sites?

Did you write wiki texts?

When have these structures been useful for you?

Did you create associations between files before you had gnowsis?

If yes, how did you create these associations?

Working with Gnowsis

I am going to observe how you use gnowsis, to see further what happened in the last years. Can you "think aloud" while you are using gnowsis, telling me what you have in your mind while using the system?

From the things you have to do today, is there a task for which you already have annotations/pimo-Things? Could you please search for these elements and work on the task?

If you don't have a task ready that is annotated in gnowsis and can be done today, could you go back to a closed task that you did in the past, or an open task for the future that you may work on? (You read an interesting paper by an author whom you met at X, you could relate the author to the paper and the conference, etc, gather material for a publication, plan a meeting, …)

Searching

Open the information releated to the task you have to do.

Do some annotations

Please add some annotations.

Reflection and thinking

Is there a related task or document that would help you now? Do you organize things in a way that help you to get things done?

Figure A.5.: Long Term Interview Guide

Feedback about each gui

Give positive and negative feedback about each user interface application. We are interested in how the GUI helped you to fill/extend/benefit from your PIMO, not about the colors or buttons (we know they are not well designed). If a button missed that would have helped you achieve a task, name it. Also name features you never used and features that were crucial to you.

- Wiki
- Thing editor
- DropBox
- Miniquire
- Search
- Tagging Plugins

Game: Given I would take away only one feature, what would you miss least?

End questions

Do you have other tools that you use to manage your information?

How does gnowsis compare to bookmarking tools? Give an example how you achieve goals using gnowsis or a bookmarking tool.

How does gnowsis compare to conventional file keeping? Give an example when gnowsis was different than conventional file keeping.

The PIMO structures you created, did they help you in the way you expected it when you did create them? Where there positive effects you didn't think of before (serendipity) or negative effects where the structures didn't help you but you thought they would?

Given we would take away gnowsis from you today, what would change and how would it affect you?

What would you miss most?

How did you create your PIMO, did it evolve over time, did you create it automatically? Do you have special tools to create it?

In which regards is your personal information model important to you? Compare it to your filesystem.

Given your PIMO is taken away, what alternatives would you use to simulate it? Which tasks that would take longer if we take away your PIMO, which won't change at all?

(The investement of creating the PIMO, was it worth it?)

At the end of the interview

Can I copy you gnowsis-activity log file to analyze it? I will anonymize it, keep it confidential.

Figure A.6.: Long Term Interview Guide

(If absolutely no, send you a software that does the analyzation?)

Thanks for the interview, I will analyze the data.
Can I come back to you in a week to ask some open questions, if some arrive?

After the Interview

Analysis of the interview in comparison to the claim of gnowsis to capture a personal information model

Ask again on some open questions.

Figure A.7.: Long Term Interview Guide

A.6. Expectation Interview 2008

For the expectation interviews from Chapter 10, four documents were used: a consent form for participants, an information flyer, and two interview guides.

NEPOMUK Semantic Desktop Studie 2008

Einverständniserklärung

Ich bestätige dass ich über 18 Jahre alt bin und an einer Studie der TU Kaiserslautern und DFKI GmbH Abteilung Wissensmanagement, durchgeführt von Hr. Leo Sauermann, teilnehmen möchte.

Ziel der Studie ist es, Erkenntnisse über das Persönliche Informationsmanagement der Teilnehmer zu sammlen. Dies betrifft das Speichern, Finden, und Organisieren von Information in digitaler Form, etwa E-Mails, Dokumente, Termine, oder Web-Links. Das NEPOMUK Projekt hat einen Prototyp entwickelt der diese Aktivitäten erleichert und verbessert, dieser soll bewertet und verbessert werden.

Im Rahmen der Studie werde ich Fragen über die Ordnerstruktur für Dateien, E-Mails, Notizen, und Web-Links beantworten. Die Namen der Ordner werden erfasst, der Inhalt von Dateien oder E-Mails wird nie betrachtet. Teil der Studie ist es auch, Aufgaben aus meiner täglichen Arbeit zu erledigen. Ich werde auch den NEPOMUK Prototyp testen.

Alle Daten die in dieser Studie erfasst werden sind vertraulich und mein Name wird nicht mit meinen Daten verbunden. Die Ergebnisse der Studie sind Eigentum der Wissenschafter.

Ich verstehe dass die Teilnahme freiwillig ist und ich jederzeit Fragen stellen kann und meine Teilnahme jederzeit beenden kann.

_____
Name des Teilnehmer (Grossbuchstaben)

_____ _____
Unterschrift des Teilnehmers Datum, Ort

_____ _____
Unterschrift des Wissenschafters Datum, Ort

Kontakt: Leo Sauermann, leo.sauermann@dfki.de
Information: http://gnowsis.opendfki.de/wiki/NepomukEvaluation2008

Figure A.8.: Consent Form

NEPOMUK Semantic Desktop Studie 2008

Wie können wir in Zukunft Information verwalten und finden? NEPOMUK ist ein Europäisches Forschungsprojekt dass den „Social Semantic Desktop" entwickelt und Erkenntnisse über Persönliches Informations-Management sammelt.

Zukünftige Betriebssysteme werden Daten nach Konzepten ordnen und finden können. Dazu werden „Ontologien" eingesetzt, Strukturen die dem menschlichen Mentalen Modell ähnlicher sind als Ordnerstrukturen in heutigen Dateisystemen. Persönlich an den Benutzer angepasst, und vom Benutzer kontrolliert, werden Personen, Projekte, Themen, Orte, Zeit, und Aufgaben als neue Ordnerstruktur verwendet. Diese Strukturen bestehen parallel zu dem bestehenden Systemen, anstatt bestehende Software zu ersetzen wird diese erweitert. Suche funktioniert dann nicht nur über den Volltext (wie bei Google Desktop) sondern über Verknüpfungen und die Beziehungen der Dateien. Es ist nun möglich, von einem Termin zu verknüpften Personen oder Projekten zu „browsen", von einer Anwendung zur anderen. Das Betriebsystem denkt vernetzt, genauso wie der Benutzer.

The Social Semantic Desktop
NEPOMUK

NEPOMUK ist ein EU-IP Projekt mit 16 Partnern,
Kontakt: DFKI Kaiserslautern
Projektlaufzeit: 2006-2008
Budget: 16 mio €
Web:
nepomuk.semanticdesktop.org

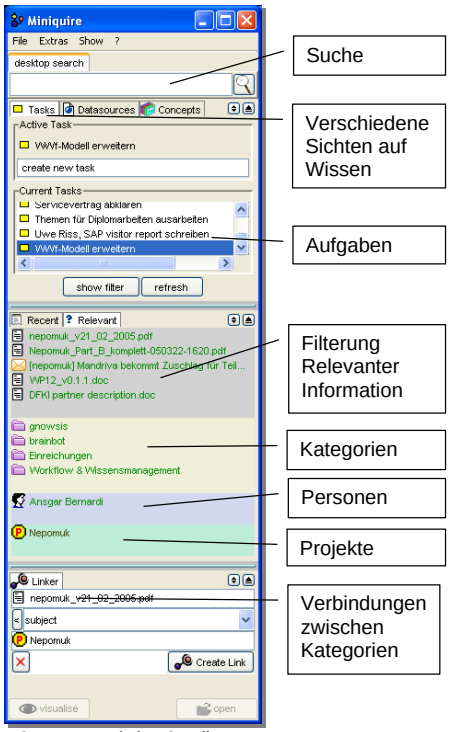
Leo Sauermann ist Wissenschafter am DFKI und Verantwortlich für den Persönlichen Semantic Desktop. Er leitet diese Studie.
www.dfki.de/~sauermann

In der Studie analysieren wir die Situation von Wissensarbeitern in Deutschen Unternehmen im Kreis Kaiserslautern (dem Standort der DFKI Semantic Desktop gruppe). Wir suchen **15 Wissensarbeiter** die in folgenden Aufgabenstellungen tätig sind:

- Beantwortung von E-Mails ohne vorgefertigte Antworten zu haben.
- Recherche von Information im Internet, Ablage der gefundenen Web-Seiten am Desktop PC.
- Organisation von Meetings oder Teilnahme an Meetings, Vorbereitung von Präsentationen für Meetings und Verbreitung dieser.
- Unstrukturierte Projektarbeit, selbständige und verantwortliche Tätigkeiten und Aufgaben (keine Standard Workflows, sondern ad-hoc workflows)

Ihr Aufgabengebiet ist also nicht komplett automatisiert, sondern erfordert Kreativität und Selbst-Organisation. Mitarbeiter deren tägliche Tätigkeit bereits komplett durch Software unterstützt wird sind **nicht Teil der Studie**. Generell suchen wir Mitarbeiter von KMU Unternehmen und Selbständige im Raum Kaiserslautern, die Branchen sind nicht relevant.

Figure A.9.: Information Flyer 1/2



Gegenstand der Studie:
Persönliches Informationsmanagmeent

Im Rahmen der Studie werden folgende Untersuchungen durchgeführt:

- Ein Interview über Ihre bestehende Methodik für das Persönliche Wissensmanagement: wie verwenden sie Folder, wie organisieren sie E-Mails, welche Strategien zur Suche verwenden sie, welche Software verwenden sie, wo haben sie Probleme.
- Vorstellung einer Software zum Informationsmanagement auf einem Demo-Laptop des DFKI. Sie führen etwa drei Aufgaben auf diesem neuen System durch.
- Fragen, welche Probleme ihres Persönlichen Informations-Managements das neue System lösen kann.

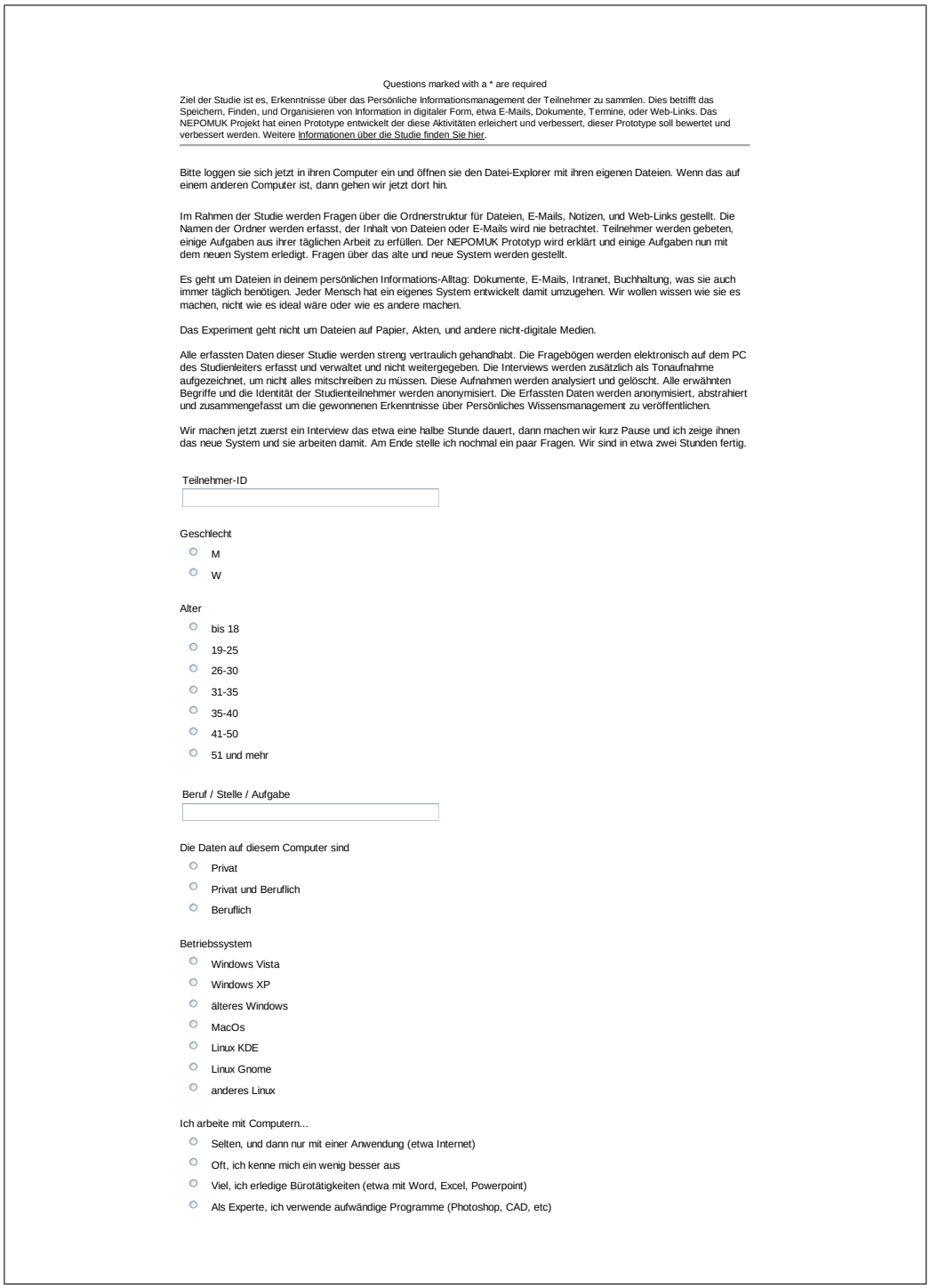
Der Zeitaufwand ist etwa zwei Stunden und wird entlohnt, mit etwa 10EUR pro Stunde. Die Studie wird im Oktober-November 2008 durchgeführt.

Vertraulichkeit

Alle erfassten Daten dieser Studie werden streng vertraulich gehandhabt.
Die Fragebögen werden elektronisch auf dem PC des Studienleiters erfasst und verwaltet und nicht weitergegeben.
Alle erwähnten Begriffe und die Identität der Studienteilnehmer werden anonymisiert. Die Erfassten Daten werden anonymisiert, abstrahiert und zusammengefasst um die gewonnenen Erkenntnisse über Persönliches Wissensmanagement zu veröffentlichen.

Kontakt: Leo Sauermann, leo.sauermann@dfki.de

Figure A.10.: Information Flyer 2/2



Questions marked with a * are required

Ziel der Studie ist es, Erkenntnisse über das Persönliche Informationsmanagement der Teilnehmer zu sammlen. Dies betrifft das Speichern, Finden, und Organisieren von Information in digitaler Form, etwa E-Mails, Dokumente, Termine, oder Web-Links. Das NEPOMUK Projekt hat einen Prototype entwickelt der diese Aktivitäten erleichert und verbessert, dieser Prototype soll bewertet und verbessert werden. Weitere Informationen über die Studie finden Sie hier.

Bitte loggen sie sich jetzt in ihren Computer ein und öffnen sie den Datei-Explorer mit ihren eigenen Dateien. Wenn das auf einem anderen Computer ist, dann gehen wir jetzt dort hin.

Im Rahmen der Studie werden Fragen über die Ordnerstruktur für Dateien, E-Mails, Notizen, und Web-Links gestellt. Die Namen der Ordner werden erfasst, der Inhalt von Dateien oder E-Mails wird nie betrachtet. Teilnehmer werden gebeten, einige Aufgaben aus ihrer täglichen Arbeit zu erfüllen. Der NEPOMUK Prototyp wird erklärt und einige Aufgaben nun mit dem neuen System erledigt. Fragen über das alte und neue System werden gestellt.

Es geht um Dateien in deinem persönlichen Informations-Alltag: Dokumente, E-Mails, Intranet, Buchhaltung, was sie auch immer täglich benötigen. Jeder Mensch hat ein eigenes System entwickelt damit umzugehen. Wir wollen wissen wie sie es machen, nicht wie es ideal wäre oder wie es andere machen.

Das Experiment geht nicht um Dateien auf Papier, Akten, und andere nicht-digitale Medien.

Alle erfassten Daten dieser Studie werden streng vertraulich gehandhabt. Die Fragebögen werden elektronisch auf dem PC des Studienleiters erfasst und verwaltet und nicht weitergegeben. Die Interviews werden zusätzlich als Tonaufnahme aufgezeichnet, um nicht alles mitschreiben zu müssen. Diese Aufnahmen werden analysiert und gelöscht. Alle erwähnten Begriffe und die Identität der Studienteilnehmer werden anonymisiert. Die Erfassten Daten werden anonymisiert, abstrahiert und zusammengefasst um die gewonnenen Erkenntnisse über Persönliches Wissensmanagement zu veröffentlichen.

Wir machen jetzt zuerst ein Interview das etwa eine halbe Stunde dauert, dann machen wir kurz Pause und ich zeige ihnen das neue System und sie arbeiten damit. Am Ende stelle ich nochmal ein paar Fragen. Wir sind in etwa zwei Stunden fertig.

Teilnehmer-ID

Geschlecht
- M
- W

Alter
- bis 18
- 19-25
- 26-30
- 31-35
- 35-40
- 41-50
- 51 und mehr

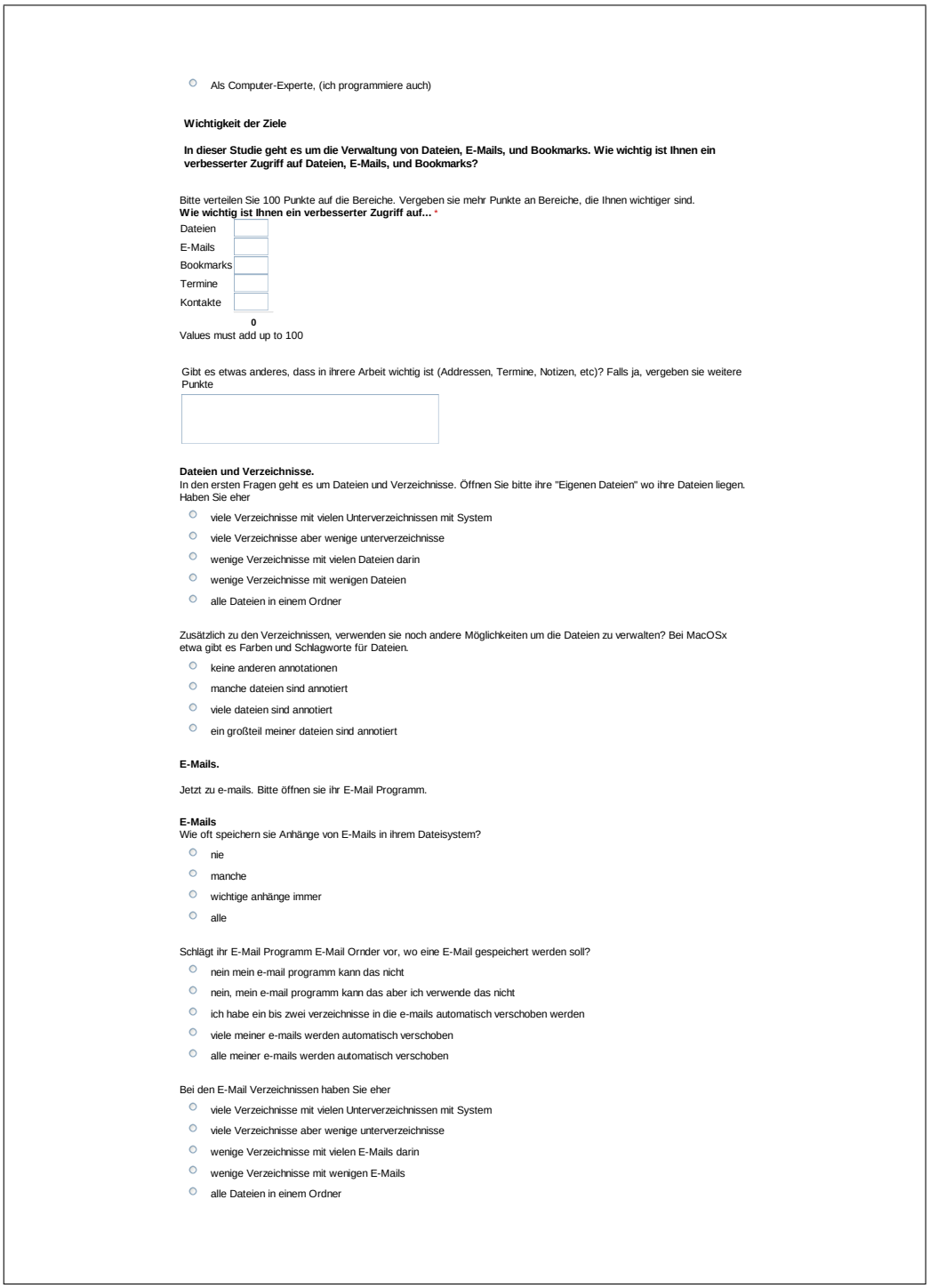
Beruf / Stelle / Aufgabe

Die Daten auf diesem Computer sind
- Privat
- Privat und Beruflich
- Beruflich

Betriebssystem
- Windows Vista
- Windows XP
- älteres Windows
- MacOs
- Linux KDE
- Linux Gnome
- anderes Linux

Ich arbeite mit Computern...
- Selten, und dann nur mit einer Anwendung (etwa Internet)
- Oft, ich kenne mich ein wenig besser aus
- Viel, ich erledige Bürotätigkeiten (etwa mit Word, Excel, Powerpoint)
- Als Experte, ich verwende aufwändige Programme (Photoshop, CAD, etc)

Figure A.11.: 2008 Expectation Questionnaire Part I, 1/9



- Als Computer-Experte, (ich programmiere auch)

Wichtigkeit der Ziele

In dieser Studie geht es um die Verwaltung von Dateien, E-Mails, und Bookmarks. Wie wichtig ist Ihnen ein verbesserter Zugriff auf Dateien, E-Mails, und Bookmarks?

Bitte verteilen Sie 100 Punkte auf die Bereiche. Vergeben sie mehr Punkte an Bereiche, die Ihnen wichtiger sind.
Wie wichtig ist Ihnen ein verbesserter Zugriff auf... *

Dateien	
E-Mails	
Bookmarks	
Termine	
Kontakte	
	0

Values must add up to 100

Gibt es etwas anderes, dass in ihrere Arbeit wichtig ist (Addressen, Termine, Notizen, etc)? Falls ja, vergeben sie weitere Punkte

Dateien und Verzeichnisse.
In den ersten Fragen geht es um Dateien und Verzeichnisse. Öffnen Sie bitte ihre "Eigenen Dateien" wo ihre Dateien liegen.
Haben Sie eher

- viele Verzeichnisse mit vielen Unterverzeichnissen mit System
- viele Verzeichnisse aber wenige unterverzeichnisse
- wenige Verzeichnisse mit vielen Dateien darin
- wenige Verzeichnisse mit wenigen Dateien
- alle Dateien in einem Ordner

Zusätzlich zu den Verzeichnissen, verwenden sie noch andere Möglichkeiten um die Dateien zu verwalten? Bei MacOSx etwa gibt es Farben und Schlagworte für Dateien.

- keine anderen annotationen
- manche dateien sind annotiert
- viele dateien sind annotiert
- ein großteil meiner dateien sind annotiert

E-Mails.

Jetzt zu e-mails. Bitte öffnen sie ihr E-Mail Programm.

E-Mails
Wie oft speichern sie Anhänge von E-Mails in ihrem Dateisystem?

- nie
- manche
- wichtige anhänge immer
- alle

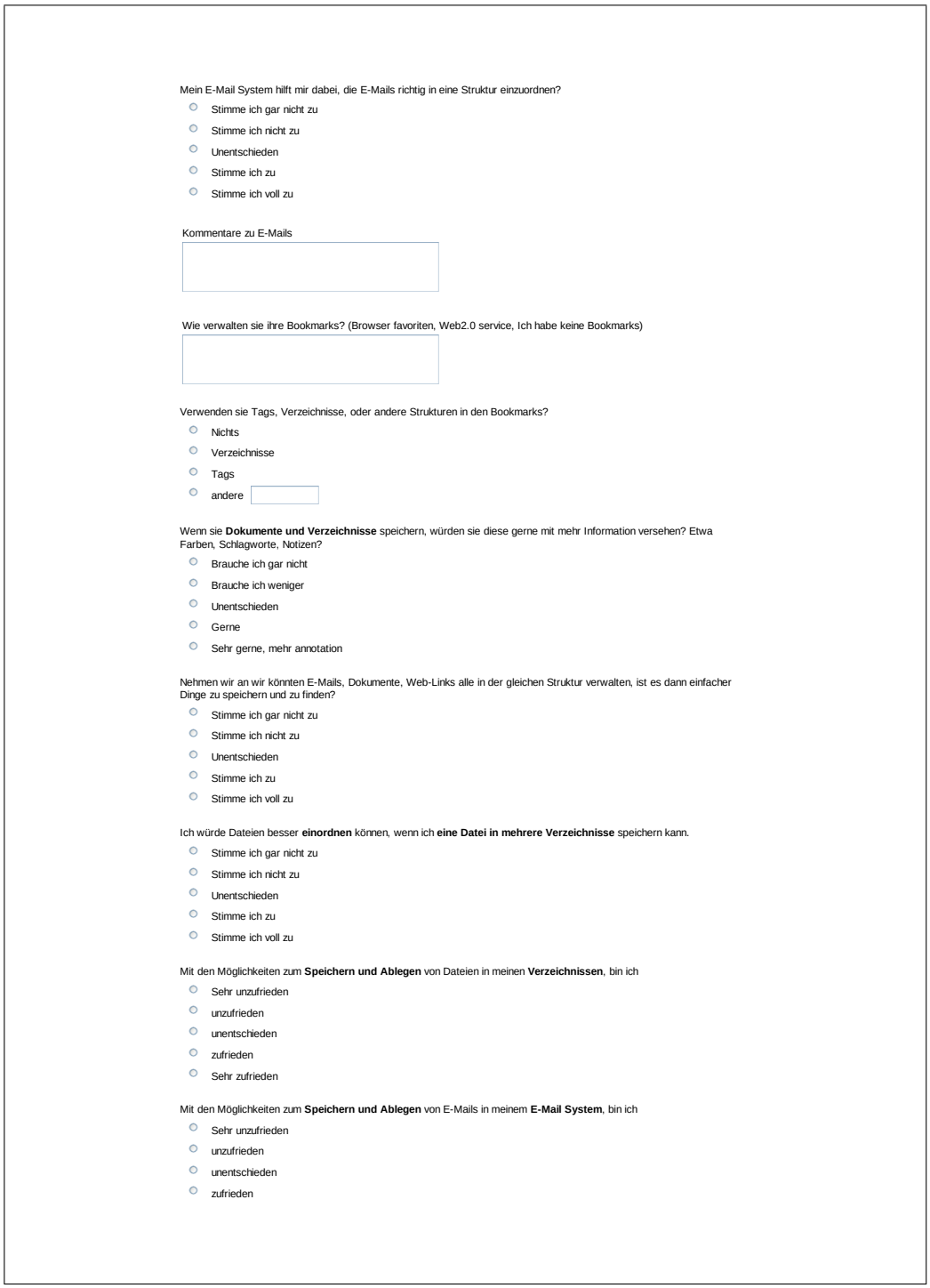
Schlägt ihr E-Mail Programm E-Mail Ornder vor, wo eine E-Mail gespeichert werden soll?

- nein mein e-mail programm kann das nicht
- nein, mein e-mail programm kann das aber ich verwende das nicht
- ich habe ein bis zwei verzeichnisse in die e-mails automatisch verschoben werden
- viele meiner e-mails werden automatisch verschoben
- alle meiner e-mails werden automatisch verschoben

Bei den E-Mail Verzeichnissen haben Sie eher

- viele Verzeichnisse mit vielen Unterverzeichnissen mit System
- viele Verzeichnisse aber wenige unterverzeichnisse
- wenige Verzeichnisse mit vielen E-Mails darin
- wenige Verzeichnisse mit wenigen E-Mails
- alle Dateien in einem Ordner

Figure A.12.: 2008 Expectation Questionnaire Part I, 2/9



Mein E-Mail System hilft mir dabei, die E-Mails richtig in eine Struktur einzuordnen?

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Kommentare zu E-Mails

Wie verwalten sie ihre Bookmarks? (Browser favoriten, Web2.0 service, Ich habe keine Bookmarks)

Verwenden sie Tags, Verzeichnisse, oder andere Strukturen in den Bookmarks?

- Nichts
- Verzeichnisse
- Tags
- andere

Wenn sie **Dokumente und Verzeichnisse** speichern, würden sie diese gerne mit mehr Information versehen? Etwa Farben, Schlagworte, Notizen?

- Brauche ich gar nicht
- Brauche ich weniger
- Unentschieden
- Gerne
- Sehr gerne, mehr annotation

Nehmen wir an wir könnten E-Mails, Dokumente, Web-Links alle in der gleichen Struktur verwalten, ist es dann einfacher Dinge zu speichern und zu finden?

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Ich würde Dateien besser **einordnen** können, wenn ich **eine Datei in mehrere Verzeichnisse** speichern kann.

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

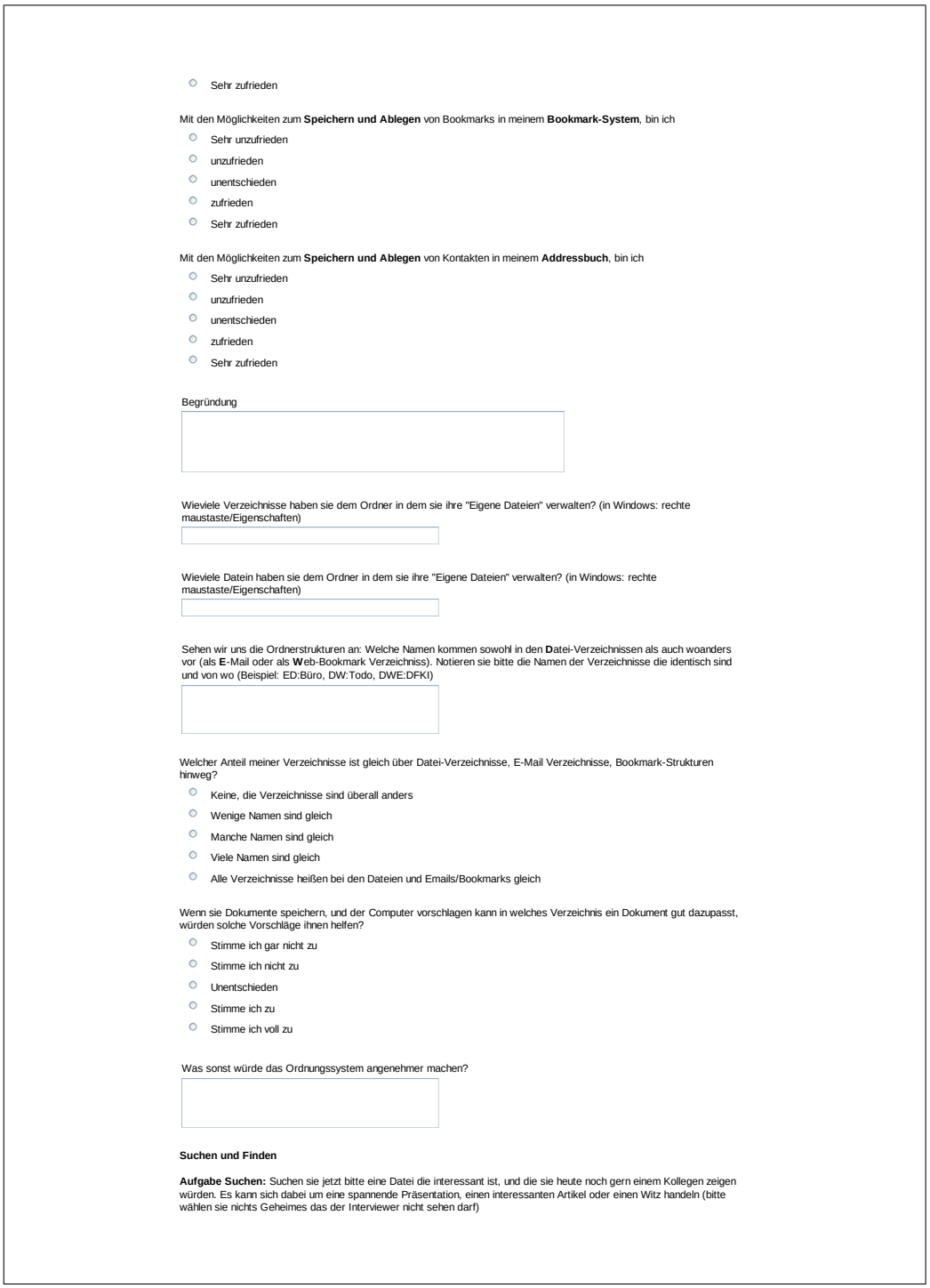
Mit den Möglichkeiten zum **Speichern und Ablegen** von Dateien in meinen **Verzeichnissen**, bin ich

- Sehr unzufrieden
- unzufrieden
- unentschieden
- zufrieden
- Sehr zufrieden

Mit den Möglichkeiten zum **Speichern und Ablegen** von E-Mails in meinem **E-Mail System**, bin ich

- Sehr unzufrieden
- unzufrieden
- unentschieden
- zufrieden

Figure A.13.: 2008 Expectation Questionnaire Part I, 3/9



- Sehr zufrieden

Mit den Möglichkeiten zum **Speichern und Ablegen** von Bookmarks in meinem **Bookmark-System**, bin ich

- Sehr unzufrieden
- unzufrieden
- unentschieden
- zufrieden
- Sehr zufrieden

Mit den Möglichkeiten zum **Speichern und Ablegen** von Kontakten in meinem **Addressbuch**, bin ich

- Sehr unzufrieden
- unzufrieden
- unentschieden
- zufrieden
- Sehr zufrieden

Begründung

Wieviele Verzeichnisse haben sie dem Ordner in dem sie ihre "Eigene Dateien" verwalten? (in Windows: rechte maustaste/Eigenschaften)

Wieviele Datein haben sie dem Ordner in dem sie ihre "Eigene Dateien" verwalten? (in Windows: rechte maustaste/Eigenschaften)

Sehen wir uns die Ordnerstrukturen an: Welche Namen kommen sowohl in den **D**atei-Verzeichnissen als auch woanders vor (als **E**-Mail oder als **W**eb-Bookmark Verzeichniss). Notieren sie bitte die Namen der Verzeichnisse die identisch sind und von wo (Beispiel: ED:Büro, DW:Todo, DWE:DFKI)

Welcher Anteil meiner Verzeichnisse ist gleich über Datei-Verzeichnisse, E-Mail Verzeichnisse, Bookmark-Strukturen hinweg?

- Keine, die Verzeichnisse sind überall anders
- Wenige Namen sind gleich
- Manche Namen sind gleich
- Viele Namen sind gleich
- Alle Verzeichnisse heißen bei den Dateien und Emails/Bookmarks gleich

Wenn sie Dokumente speichern, und der Computer vorschlagen kann in welches Verzeichnis ein Dokument gut dazupasst, würden solche Vorschläge ihnen helfen?

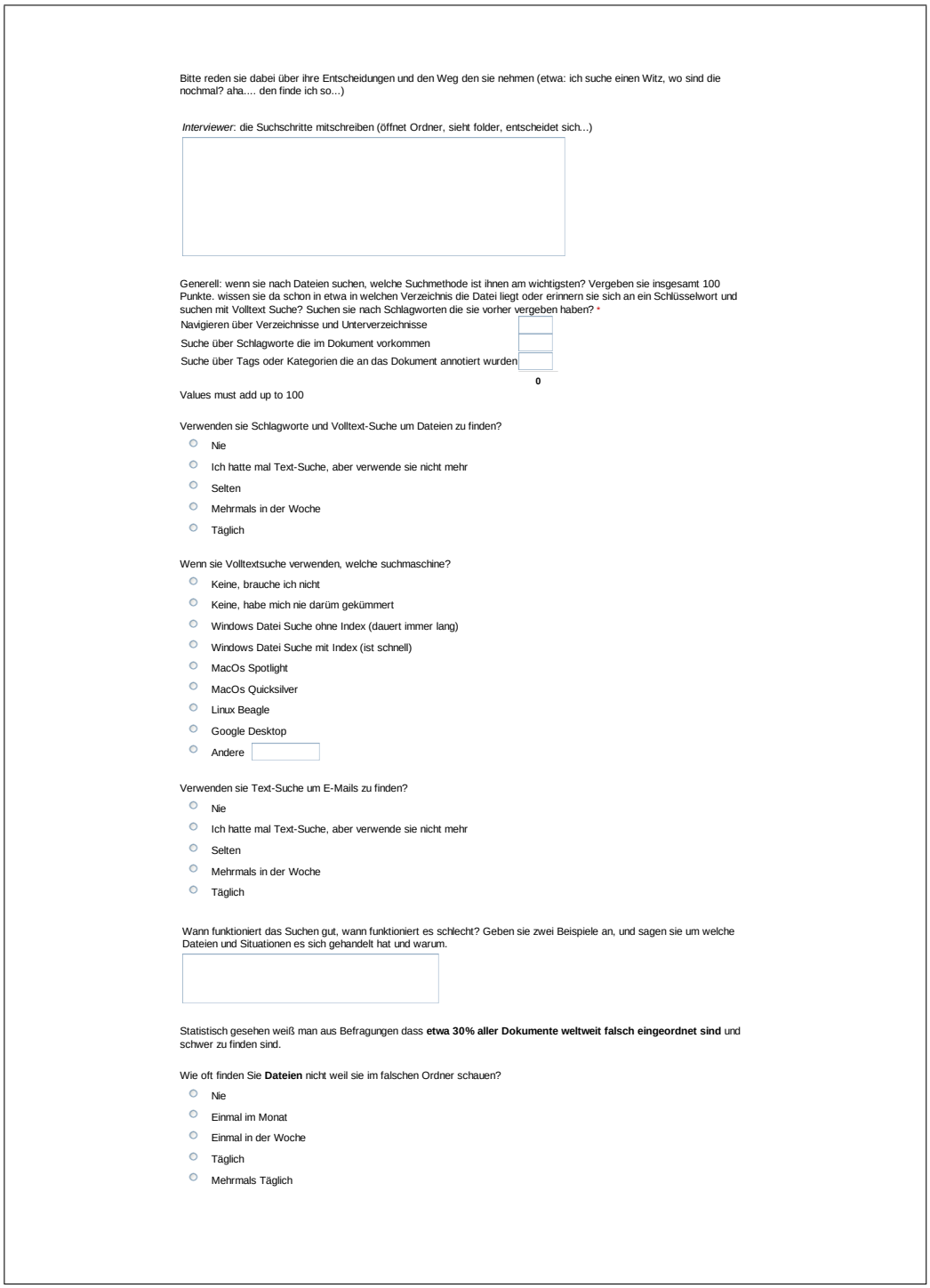
- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Was sonst würde das Ordnungssystem angenehmer machen?

Suchen und Finden

Aufgabe Suchen: Suchen sie jetzt bitte eine Datei die interessant ist, und die sie heute noch gern einem Kollegen zeigen würden. Es kann sich dabei um eine spannende Präsentation, einen interessanten Artikel oder einen Witz handeln (bitte wählen sie nichts Geheimes das der Interviewer nicht sehen darf)

Figure A.14.: 2008 Expectation Questionnaire Part I, 4/9



Bitte reden sie dabei über ihre Entscheidungen und den Weg den sie nehmen (etwa: ich suche einen Witz, wo sind die nochmal? aha.... den finde ich so...)

Interviewer: die Suchschritte mitschreiben (öffnet Ordner, sieht folder, entscheidet sich...)

Generell: wenn sie nach Dateien suchen, welche Suchmethode ist ihnen am wichtigsten? Vergeben sie insgesamt 100 Punkte. wissen sie da schon in etwa in welchen Verzeichnis die Datei liegt oder erinnern sie sich an ein Schlüsselwort und suchen mit Volltext Suche? Suchen sie nach Schlagworten die sie vorher vergeben haben? *

Navigieren über Verzeichnisse und Unterverzeichnisse

Suche über Schlagworte die im Dokument vorkommen

Suche über Tags oder Kategorien die an das Dokument annotiert wurden

0

Values must add up to 100

Verwenden sie Schlagworte und Volltext-Suche um Dateien zu finden?

- Nie
- Ich hatte mal Text-Suche, aber verwende sie nicht mehr
- Selten
- Mehrmals in der Woche
- Täglich

Wenn sie Volltextsuche verwenden, welche suchmaschine?

- Keine, brauche ich nicht
- Keine, habe mich nie darüm gekümmert
- Windows Datei Suche ohne Index (dauert immer lang)
- Windows Datei Suche mit Index (ist schnell)
- MacOs Spotlight
- MacOs Quicksilver
- Linux Beagle
- Google Desktop
- Andere

Verwenden sie Text-Suche um E-Mails zu finden?

- Nie
- Ich hatte mal Text-Suche, aber verwende sie nicht mehr
- Selten
- Mehrmals in der Woche
- Täglich

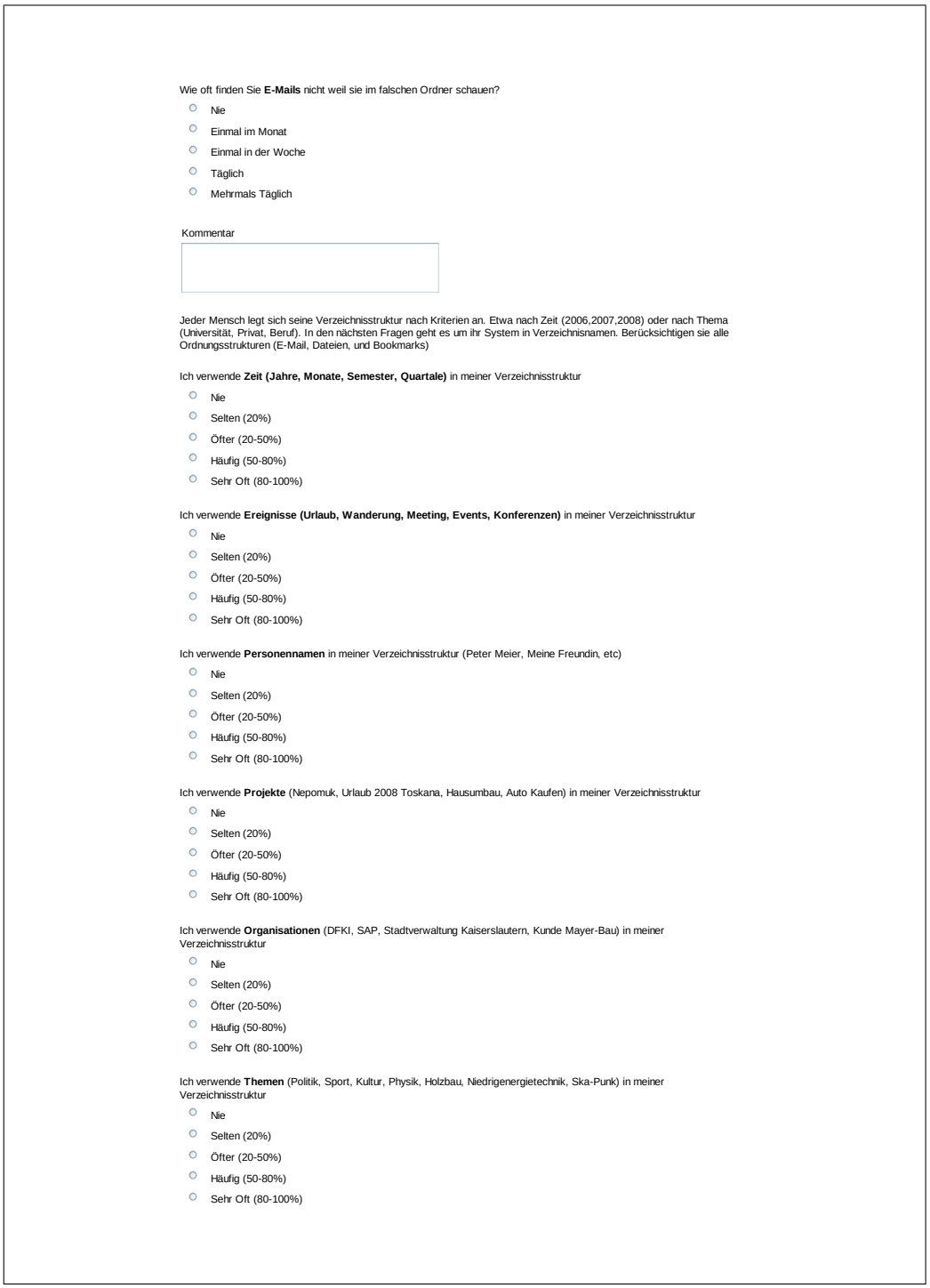
Wann funktioniert das Suchen gut, wann funktioniert es schlecht? Geben sie zwei Beispiele an, und sagen sie um welche Dateien und Situationen es sich gehandelt hat und warum.

Statistisch gesehen weiß man aus Befragungen dass **etwa 30% aller Dokumente weltweit falsch eingeordnet sind** und schwer zu finden sind.

Wie oft finden Sie **Dateien** nicht weil sie im falschen Ordner schauen?

- Nie
- Einmal im Monat
- Einmal in der Woche
- Täglich
- Mehrmals Täglich

Figure A.15.: 2008 Expectation Questionnaire Part I, 5/9



Wie oft finden Sie **E-Mails** nicht weil sie im falschen Ordner schauen?

- Nie
- Einmal im Monat
- Einmal in der Woche
- Täglich
- Mehrmals Täglich

Kommentar

Jeder Mensch legt sich seine Verzeichnisstruktur nach Kriterien an. Etwa nach Zeit (2006,2007,2008) oder nach Thema (Universität, Privat, Beruf). In den nächsten Fragen geht es um ihr System in Verzeichnisnamen. Berücksichtigen sie alle Ordnungsstrukturen (E-Mail, Dateien, und Bookmarks)

Ich verwende **Zeit (Jahre, Monate, Semester, Quartale)** in meiner Verzeichnisstruktur

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

Ich verwende **Ereignisse (Urlaub, Wanderung, Meeting, Events, Konferenzen)** in meiner Verzeichnisstruktur

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

Ich verwende **Personennamen** in meiner Verzeichnisstruktur (Peter Meier, Meine Freundin, etc)

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

Ich verwende **Projekte** (Nepomuk, Urlaub 2008 Toskana, Hausumbau, Auto Kaufen) in meiner Verzeichnisstruktur

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

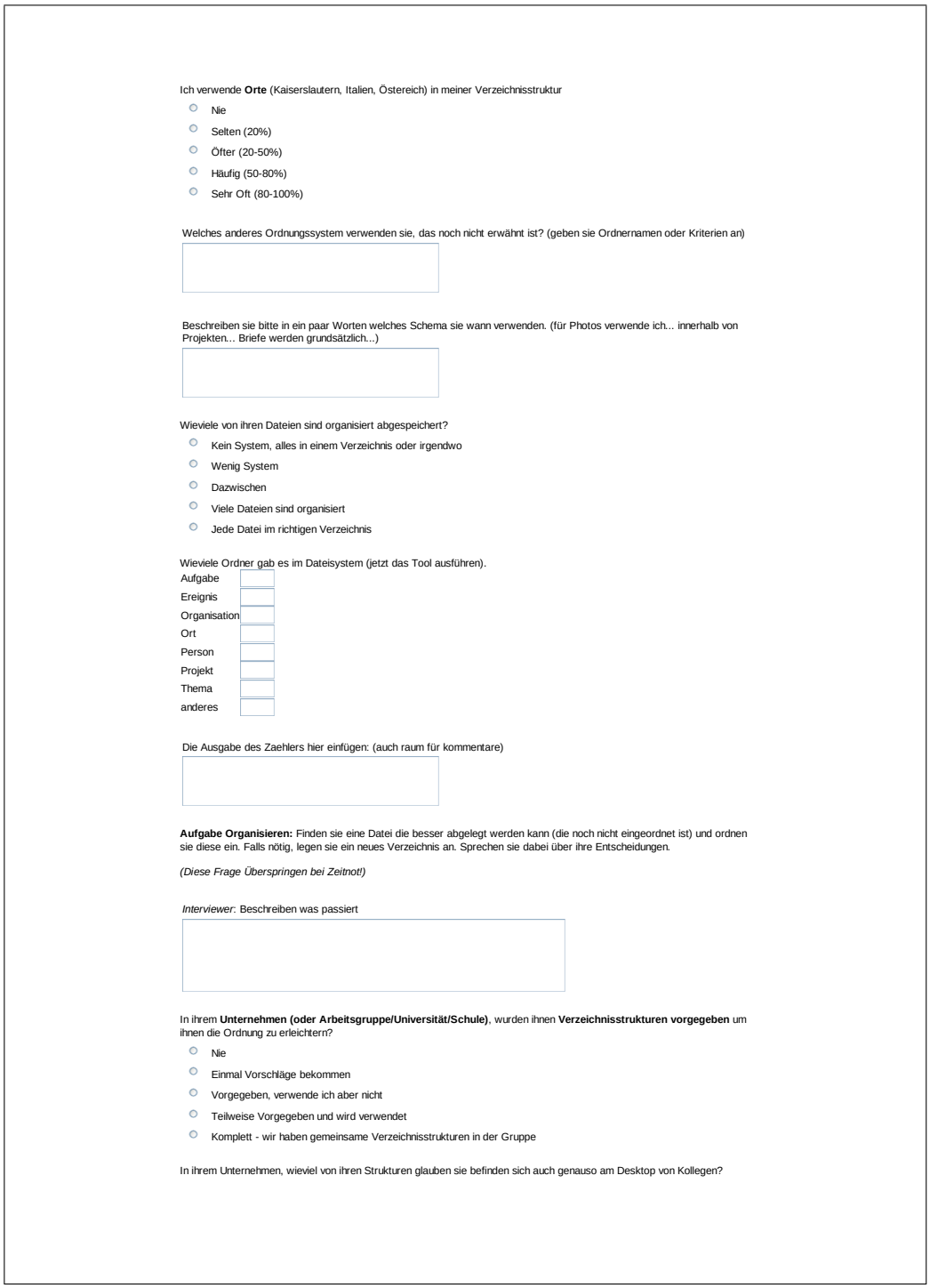
Ich verwende **Organisationen** (DFKI, SAP, Stadtverwaltung Kaiserslautern, Kunde Mayer-Bau) in meiner Verzeichnisstruktur

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

Ich verwende **Themen** (Politik, Sport, Kultur, Physik, Holzbau, Niedrigenergietechnik, Ska-Punk) in meiner Verzeichnisstruktur

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

Figure A.16.: 2008 Expectation Questionnaire Part I, 6/9



Ich verwende **Orte** (Kaiserslautern, Italien, Östereich) in meiner Verzeichnisstruktur

- Nie
- Selten (20%)
- Öfter (20-50%)
- Häufig (50-80%)
- Sehr Oft (80-100%)

Welches anderes Ordnungssystem verwenden sie, das noch nicht erwähnt ist? (geben sie Ordnernamen oder Kriterien an)

Beschreiben sie bitte in ein paar Worten welches Schema sie wann verwenden. (für Photos verwende ich... innerhalb von Projekten... Briefe werden grundsätzlich...)

Wieviele von ihren Dateien sind organisiert abgespeichert?

- Kein System, alles in einem Verzeichnis oder irgendwo
- Wenig System
- Dazwischen
- Viele Dateien sind organisiert
- Jede Datei im richtigen Verzeichnis

Wieviele Ordner gab es im Dateisystem (jetzt das Tool ausführen).

Aufgabe
Ereignis
Organisation
Ort
Person
Projekt
Thema
anderes

Die Ausgabe des Zaehlers hier einfügen: (auch raum für kommentare)

Aufgabe Organisieren: Finden sie eine Datei die besser abgelegt werden kann (die noch nicht eingeordnet ist) und ordnen sie diese ein. Falls nötig, legen sie ein neues Verzeichnis an. Sprechen sie dabei über ihre Entscheidungen.

(Diese Frage Überspringen bei Zeitnot!)

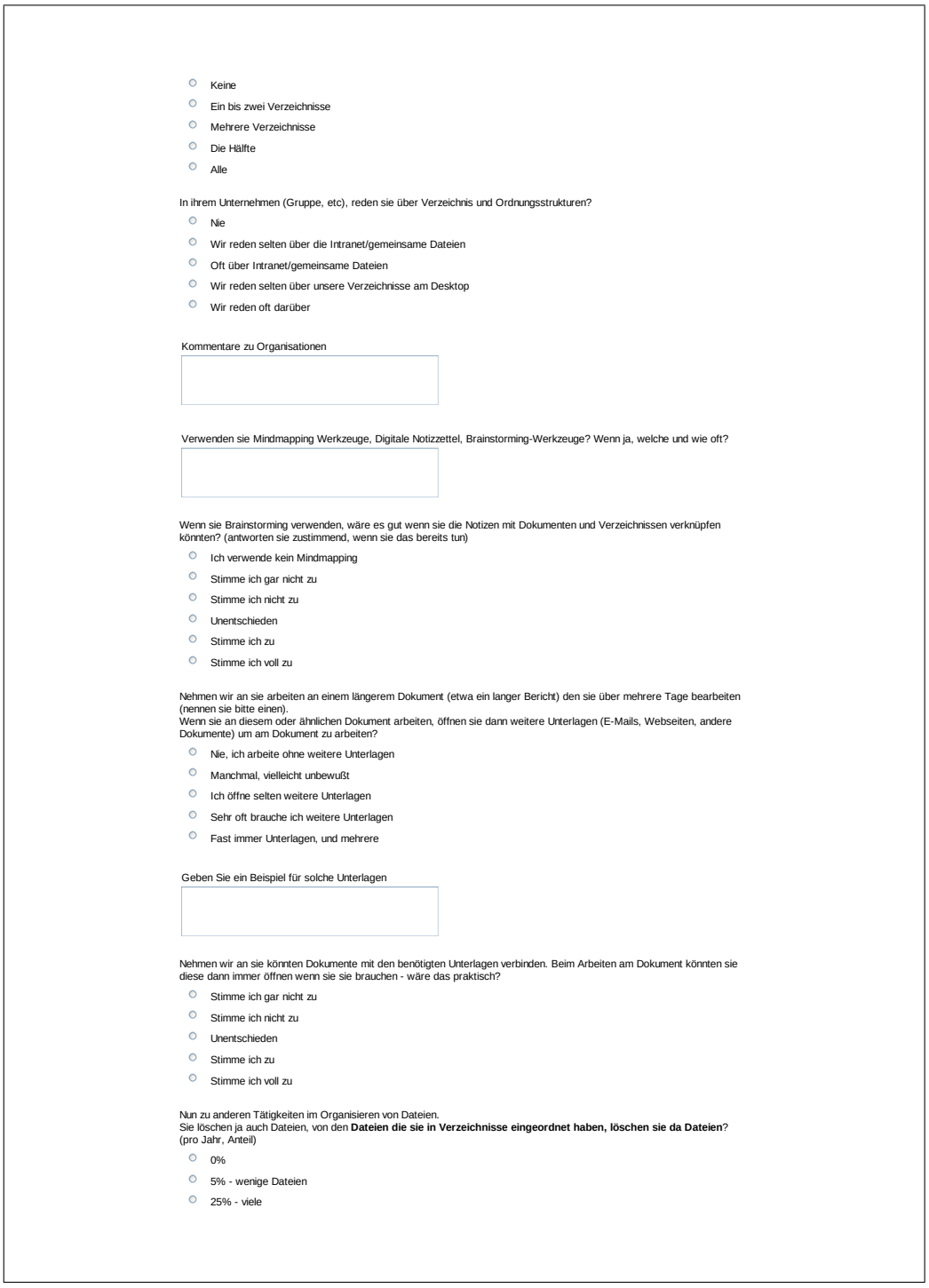
Interviewer: Beschreiben was passiert

In ihrem **Unternehmen (oder Arbeitsgruppe/Universität/Schule)**, wurden ihnen **Verzeichnisstrukturen vorgegeben** um ihnen die Ordnung zu erleichtern?

- Nie
- Einmal Vorschläge bekommen
- Vorgegeben, verwende ich aber nicht
- Teilweise Vorgegeben und wird verwendet
- Komplett - wir haben gemeinsame Verzeichnisstrukturen in der Gruppe

In ihrem Unternehmen, wieviel von ihren Strukturen glauben sie befinden sich auch genauso am Desktop von Kollegen?

Figure A.17.: 2008 Expectation Questionnaire Part I, 7/9



- Keine
- Ein bis zwei Verzeichnisse
- Mehrere Verzeichnisse
- Die Hälfte
- Alle

In ihrem Unternehmen (Gruppe, etc), reden sie über Verzeichnis und Ordnungsstrukturen?

- Nie
- Wir reden selten über die Intranet/gemeinsame Dateien
- Oft über Intranet/gemeinsame Dateien
- Wir reden selten über unsere Verzeichnisse am Desktop
- Wir reden oft darüber

Kommentare zu Organisationen

Verwenden sie Mindmapping Werkzeuge, Digitale Notizzettel, Brainstorming-Werkzeuge? Wenn ja, welche und wie oft?

Wenn sie Brainstorming verwenden, wäre es gut wenn sie die Notizen mit Dokumenten und Verzeichnissen verknüpfen könnten? (antworten sie zustimmend, wenn sie das bereits tun)

- Ich verwende kein Mindmapping
- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Nehmen wir an sie arbeiten an einem längerem Dokument (etwa ein langer Bericht) den sie über mehrere Tage bearbeiten (nennen sie bitte einen).
Wenn sie an diesem oder ähnlichen Dokument arbeiten, öffnen sie dann weitere Unterlagen (E-Mails, Webseiten, andere Dokumente) um am Dokument zu arbeiten?

- Nie, ich arbeite ohne weitere Unterlagen
- Manchmal, vielleicht unbewußt
- Ich öffne selten weitere Unterlagen
- Sehr oft brauche ich weitere Unterlagen
- Fast immer Unterlagen, und mehrere

Geben Sie ein Beispiel für solche Unterlagen

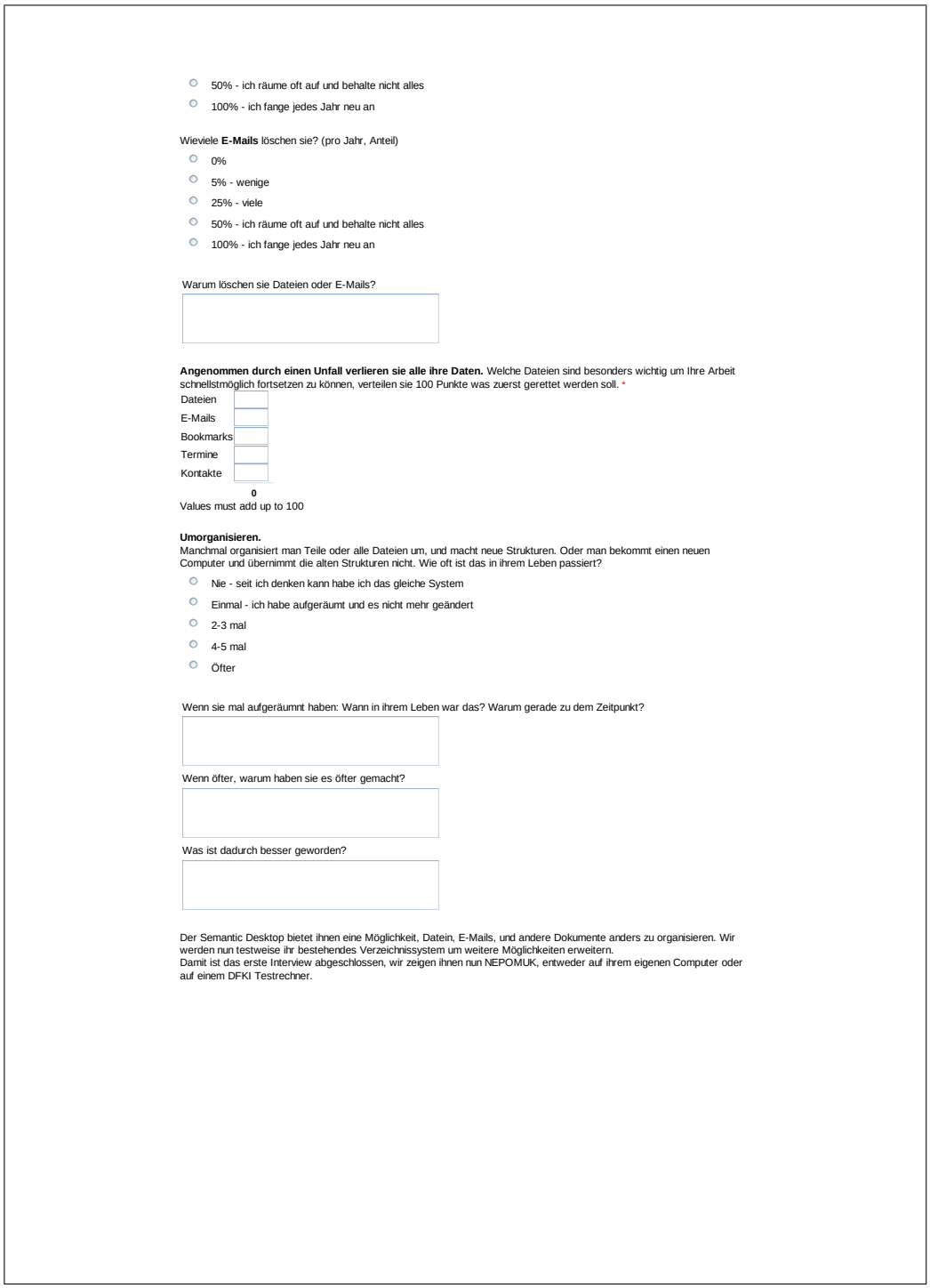
Nehmen wir an sie könnten Dokumente mit den benötigten Unterlagen verbinden. Beim Arbeiten am Dokument könnten sie diese dann immer öffnen wenn sie sie brauchen - wäre das praktisch?

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Nun zu anderen Tätigkeiten im Organisieren von Dateien.
Sie löschen ja auch Dateien, von den **Dateien die sie in Verzeichnisse eingeordnet haben, löschen sie da Dateien**?
(pro Jahr, Anteil)

- 0%
- 5% - wenige Dateien
- 25% - viele

Figure A.18.: 2008 Expectation Questionnaire Part I, 8/9



- 50% - ich räume oft auf und behalte nicht alles
- 100% - ich fange jedes Jahr neu an

Wieviele **E-Mails** löschen sie? (pro Jahr, Anteil)

- 0%
- 5% - wenige
- 25% - viele
- 50% - ich räume oft auf und behalte nicht alles
- 100% - ich fange jedes Jahr neu an

Warum löschen sie Dateien oder E-Mails?

Angenommen durch einen Unfall verlieren sie alle ihre Daten. Welche Dateien sind besonders wichtig um Ihre Arbeit schnellstmöglich fortsetzen zu können, verteilen sie 100 Punkte was zuerst gerettet werden soll. *

Dateien
E-Mails
Bookmarks
Termine
Kontakte

0

Values must add up to 100

Umorganisieren.
Manchmal organisiert man Teile oder alle Dateien um, und macht neue Strukturen. Oder man bekommt einen neuen Computer und übernimmt die alten Strukturen nicht. Wie oft ist das in ihrem Leben passiert?

- Nie - seit ich denken kann habe ich das gleiche System
- Einmal - ich habe aufgeräumt und es nicht mehr geändert
- 2-3 mal
- 4-5 mal
- Öfter

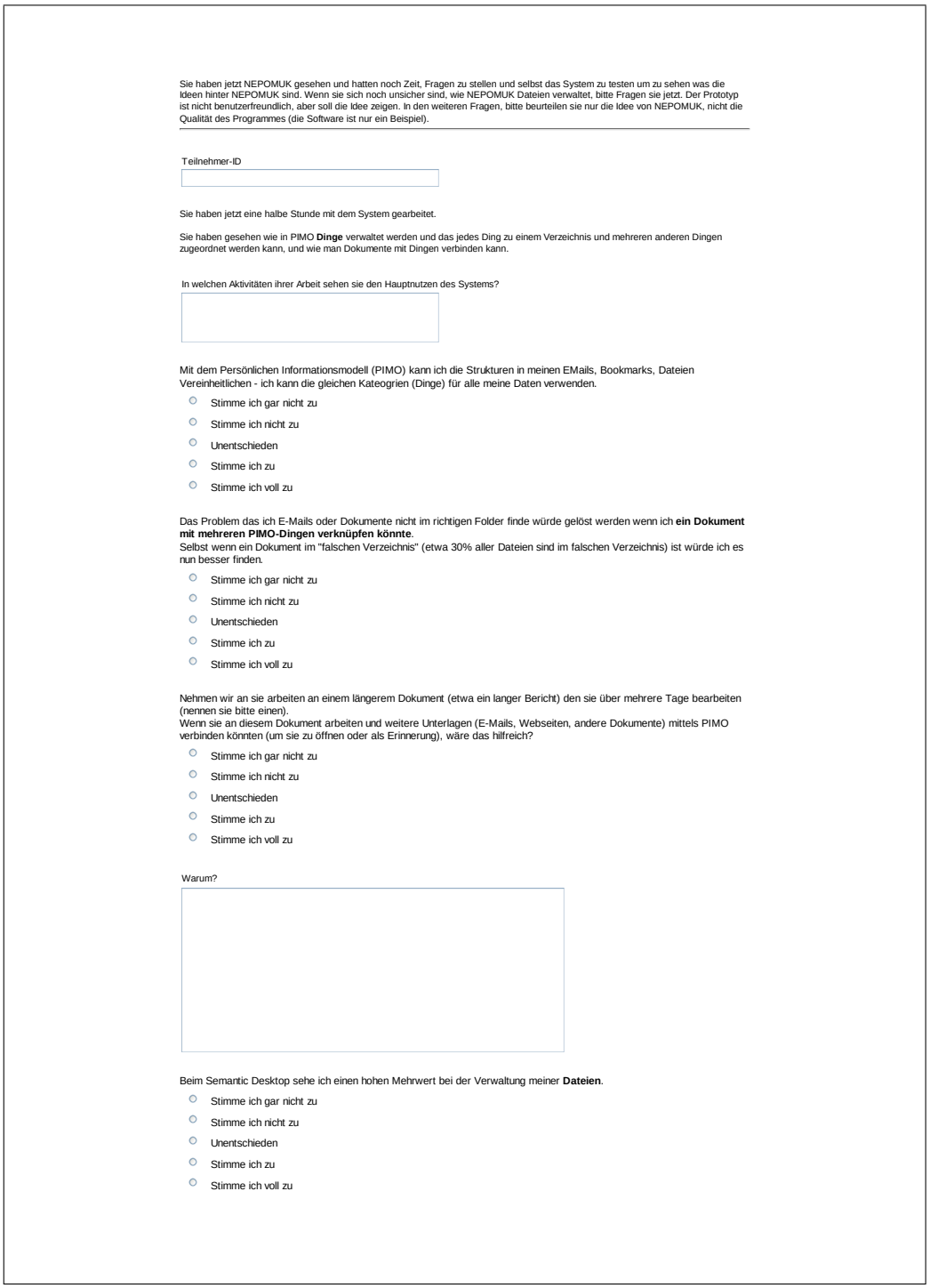
Wenn sie mal aufgeräumt haben: Wann in ihrem Leben war das? Warum gerade zu dem Zeitpunkt?

Wenn öfter, warum haben sie es öfter gemacht?

Was ist dadurch besser geworden?

Der Semantic Desktop bietet ihnen eine Möglichkeit, Datein, E-Mails, und andere Dokumente anders zu organisieren. Wir werden nun testweise ihr bestehendes Verzeichnissystem um weitere Möglichkeiten erweitern.
Damit ist das erste Interview abgeschlossen, wir zeigen ihnen nun NEPOMUK, entweder auf ihrem eigenen Computer oder auf einem DFKI Testrechner.

Figure A.19.: 2008 Expectation Questionnaire Part I, 9/9



Sie haben jetzt NEPOMUK gesehen und hatten noch Zeit, Fragen zu stellen und selbst das System zu testen um zu sehen was die Ideen hinter NEPOMUK sind. Wenn sie sich noch unsicher sind, wie NEPOMUK Dateien verwaltet, bitte Fragen sie jetzt. Der Prototyp ist nicht benutzerfreundlich, aber soll die Idee zeigen. In den weiteren Fragen, bitte beurteilen sie nur die Idee von NEPOMUK, nicht die Qualität des Programmes (die Software ist nur ein Beispiel).

Teilnehmer-ID

Sie haben jetzt eine halbe Stunde mit dem System gearbeitet.

Sie haben gesehen wie in PIMO **Dinge** verwaltet werden und das jedes Ding zu einem Verzeichnis und mehreren anderen Dingen zugeordnet werden kann, und wie man Dokumente mit Dingen verbinden kann.

In welchen Aktivitäten ihrer Arbeit sehen sie den Hauptnutzen des Systems?

Mit dem Persönlichen Informationsmodell (PIMO) kann ich die Strukturen in meinen EMails, Bookmarks, Dateien Vereinheitlichen - ich kann die gleichen Kateogrien (Dinge) für alle meine Daten verwenden.

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Das Problem das ich E-Mails oder Dokumente nicht im richtigen Folder finde würde gelöst werden wenn ich **ein Dokument mit mehreren PIMO-Dingen verknüpfen könnte**.
Selbst wenn ein Dokument im "falschen Verzeichnis" (etwa 30% aller Dateien sind im falschen Verzeichnis) ist würde ich es nun besser finden.

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Nehmen wir an sie arbeiten an einem längerem Dokument (etwa ein langer Bericht) den sie über mehrere Tage bearbeiten (nennen sie bitte einen).
Wenn sie an diesem Dokument arbeiten und weitere Unterlagen (E-Mails, Webseiten, andere Dokumente) mittels PIMO verbinden könnten (um sie zu öffnen oder als Erinnerung), wäre das hilfreich?

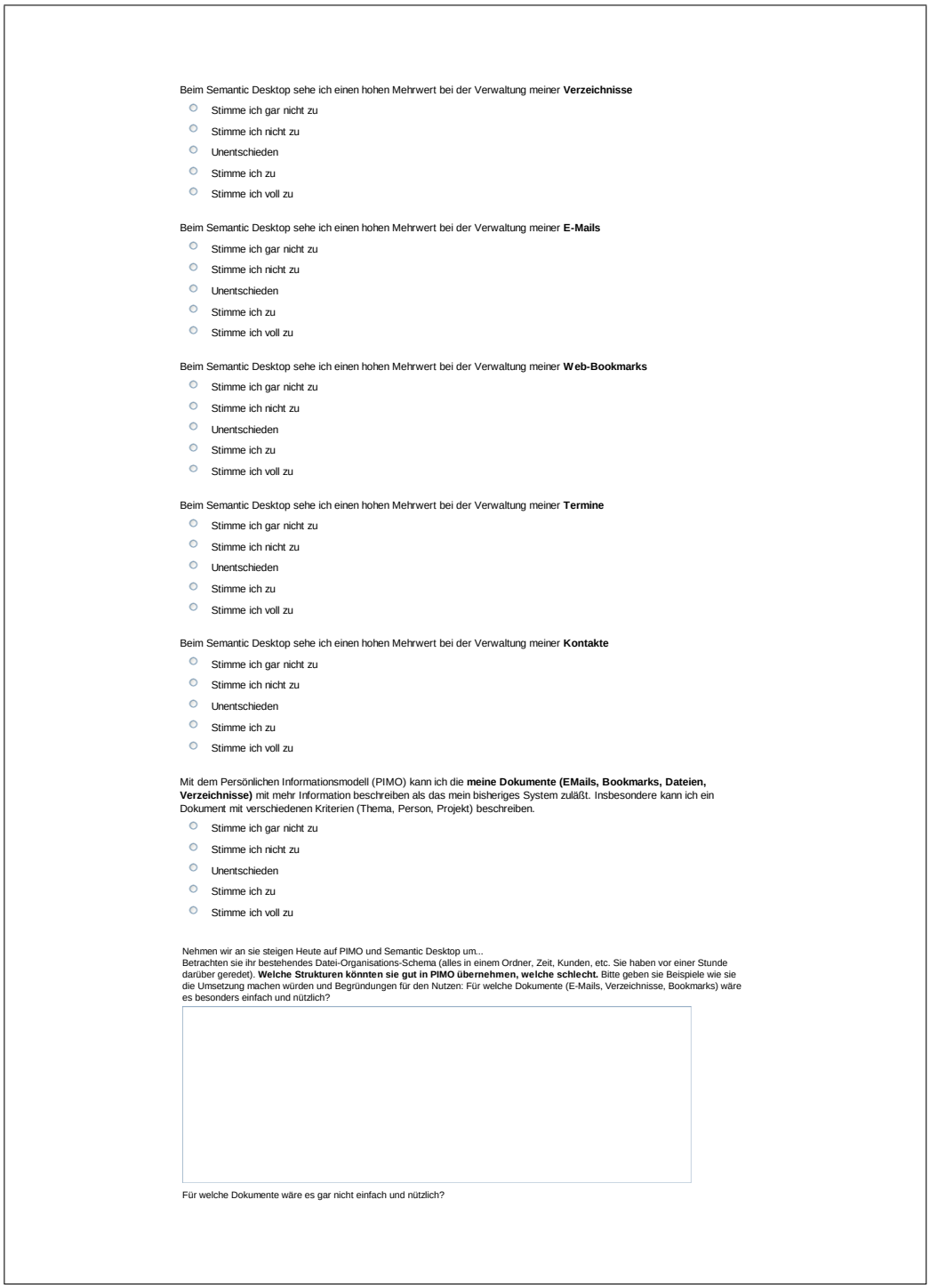
- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Warum?

Beim Semantic Desktop sehe ich einen hohen Mehrwert bei der Verwaltung meiner **Dateien**.

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Figure A.20.: 2008 Expectation Questionnaire Part II, 1/4



Beim Semantic Desktop sehe ich einen hohen Mehrwert bei der Verwaltung meiner **Verzeichnisse**

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Beim Semantic Desktop sehe ich einen hohen Mehrwert bei der Verwaltung meiner **E-Mails**

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Beim Semantic Desktop sehe ich einen hohen Mehrwert bei der Verwaltung meiner **Web-Bookmarks**

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Beim Semantic Desktop sehe ich einen hohen Mehrwert bei der Verwaltung meiner **Termine**

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Beim Semantic Desktop sehe ich einen hohen Mehrwert bei der Verwaltung meiner **Kontakte**

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

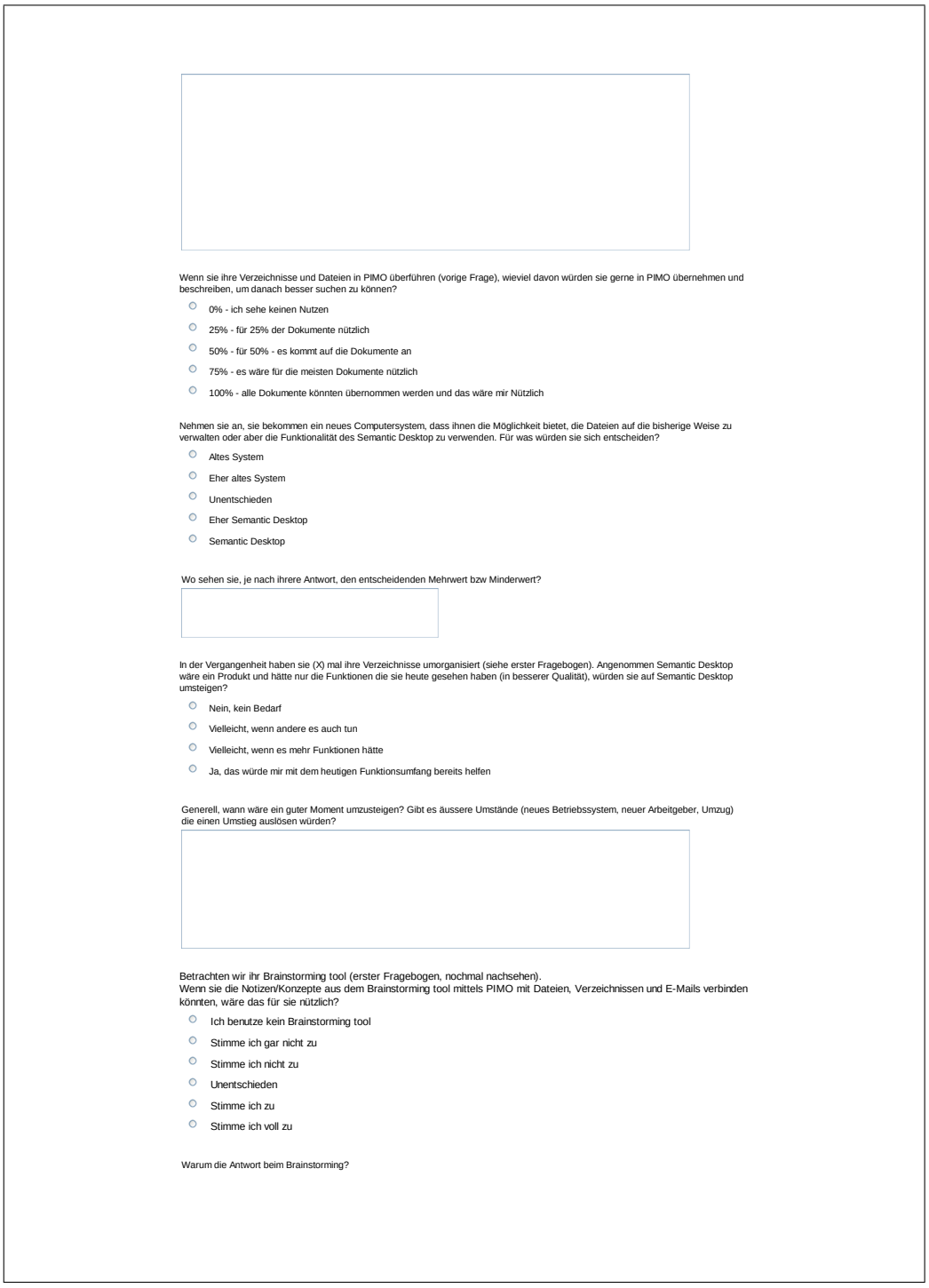
Mit dem Persönlichen Informationsmodell (PIMO) kann ich die **meine Dokumente (EMails, Bookmarks, Dateien, Verzeichnisse)** mit mehr Information beschreiben als das mein bisheriges System zuläßt. Insbesondere kann ich ein Dokument mit verschiedenen Kriterien (Thema, Person, Projekt) beschreiben.

- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Nehmen wir an sie steigen Heute auf PIMO und Semantic Desktop um...
Betrachten sie ihr bestehendes Datei-Organisations-Schema (alles in einem Ordner, Zeit, Kunden, etc. Sie haben vor einer Stunde darüber geredet). **Welche Strukturen könnten sie gut in PIMO übernehmen, welche schlecht.** Bitte geben sie Beispiele wie sie die Umsetzung machen würden und Begründungen für den Nutzen: Für welche Dokumente (E-Mails, Verzeichnisse, Bookmarks) wäre es besonders einfach und nützlich?

Für welche Dokumente wäre es gar nicht einfach und nützlich?

Figure A.21.: 2008 Expectation Questionnaire Part II, 2/4



Wenn sie ihre Verzeichnisse und Dateien in PIMO überführen (vorige Frage), wieviel davon würden sie gerne in PIMO übernehmen und beschreiben, um danach besser suchen zu können?

- 0% - ich sehe keinen Nutzen
- 25% - für 25% der Dokumente nützlich
- 50% - für 50% - es kommt auf die Dokumente an
- 75% - es wäre für die meisten Dokumente nützlich
- 100% - alle Dokumente könnten übernommen werden und das wäre mir Nützlich

Nehmen sie an, sie bekommen ein neues Computersystem, dass ihnen die Möglichkeit bietet, die Dateien auf die bisherige Weise zu verwalten oder aber die Funktionalität des Semantic Desktop zu verwenden. Für was würden sie sich entscheiden?

- Altes System
- Eher altes System
- Unentschieden
- Eher Semantic Desktop
- Semantic Desktop

Wo sehen sie, je nach ihrere Antwort, den entscheidenden Mehrwert bzw Minderwert?

In der Vergangenheit haben sie (X) mal ihre Verzeichnisse umorganisiert (siehe erster Fragebogen). Angenommen Semantic Desktop wäre ein Produkt und hätte nur die Funktionen die sie heute gesehen haben (in besserer Qualität), würden sie auf Semantic Desktop umsteigen?

- Nein, kein Bedarf
- Vielleicht, wenn andere es auch tun
- Vielleicht, wenn es mehr Funktionen hätte
- Ja, das würde mir mit dem heutigen Funktionsumfang bereits helfen

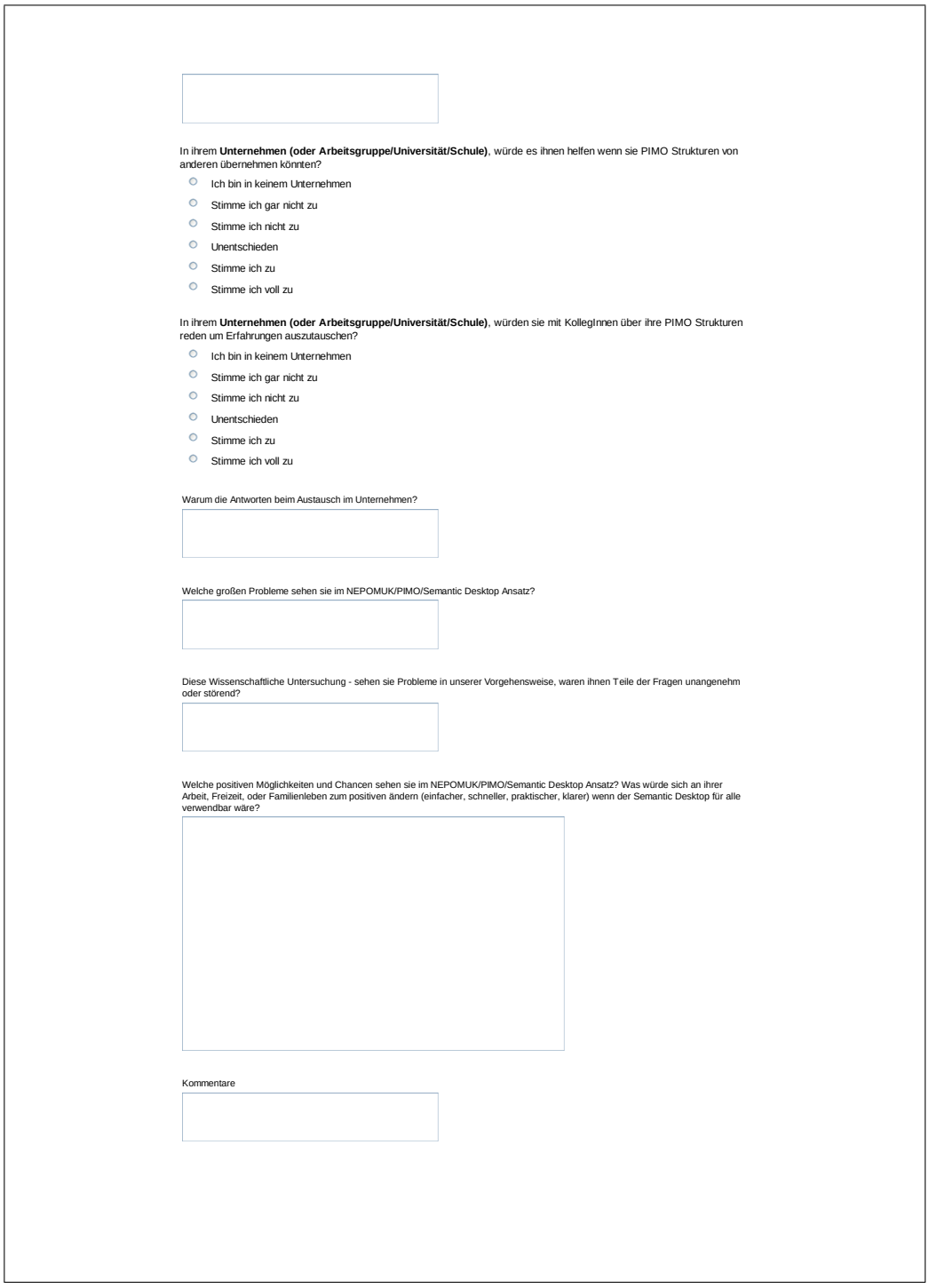
Generell, wann wäre ein guter Moment umzusteigen? Gibt es äussere Umstände (neues Betriebssystem, neuer Arbeitgeber, Umzug) die einen Umstieg auslösen würden?

Betrachten wir ihr Brainstorming tool (erster Fragebogen, nochmal nachsehen).
Wenn sie die Notizen/Konzepte aus dem Brainstorming tool mittels PIMO mit Dateien, Verzeichnissen und E-Mails verbinden könnten, wäre das für sie nützlich?

- Ich benutze kein Brainstorming tool
- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Warum die Antwort beim Brainstorming?

Figure A.22.: 2008 Expectation Questionnaire Part II, 3/4



In ihrem **Unternehmen (oder Arbeitsgruppe/Universität/Schule)**, würde es ihnen helfen wenn sie PIMO Strukturen von anderen übernehmen könnten?

- Ich bin in keinem Unternehmen
- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

In ihrem **Unternehmen (oder Arbeitsgruppe/Universität/Schule)**, würden sie mit KollegInnen über ihre PIMO Strukturen reden um Erfahrungen auszutauschen?

- Ich bin in keinem Unternehmen
- Stimme ich gar nicht zu
- Stimme ich nicht zu
- Unentschieden
- Stimme ich zu
- Stimme ich voll zu

Warum die Antworten beim Austausch im Unternehmen?

Welche großen Probleme sehen sie im NEPOMUK/PIMO/Semantic Desktop Ansatz?

Diese Wissenschaftliche Untersuchung - sehen sie Probleme in unserer Vorgehensweise, waren ihnen Teile der Fragen unangenehm oder störend?

Welche positiven Möglichkeiten und Chancen sehen sie im NEPOMUK/PIMO/Semantic Desktop Ansatz? Was würde sich an ihrer Arbeit, Freizeit, oder Familienleben zum positiven ändern (einfacher, schneller, praktischer, klarer) wenn der Semantic Desktop für alle verwendbar wäre?

Kommentare

Figure A.23.: 2008 Expectation Questionnaire Part II, 4/4

Curriculum Vitae

Personal Data		Leo Sauermann Pirmasenser Strasse 18 67655 Kaiserslautern * 30. 08. 1977 in Wien, Österreich. married to Ingrid Brunner-Sauermann
Education		
2004		Best Diploma Thesis of the Year at the faculty
1996	– 2004	Studied Information Science at the Vienna University of Technology http://www.tu-wien.ac.at
1987	– 1995	High-School GRG19, Billrothstrasse, Wien 19 http://www.grg19.asn-wien.ac.at
Military Service		
1996/01	– 1996/08	Versorgungskompanie, Radetzky Kaserne
Employment		
2004/07	– today	DFKI GmbH, Researcher
2005	– 2006	Lecturer, Semantic Web School Austria
1998/03	– 2003/02	Software Architect and Lead Programmer at Impact Business Computing http://www.impact.co.at
1999/01	– 1999/10	Software developer for Innovative Computer Factory http://www.foen-x.at
1999		Trainer for Borland Delphi courses, Apex http://www.apex4you.at
1999/07	– 2000/03	Webmaster of www.schulprofi.at, a learning website for school kids
1998		Compaq promotion team. Sales of pcs and laptops.
1997/04	– 1998/06	Borland Delphi programmer for Kurt Vesely Software company, Vienna. My work included recruiting and leading a software development team.
1996		Helping hand in the musical production “Cabaret”, Vienna

Photo by Frank Steinmann

I finished my information science studies with the award-winning diploma thesis “gnowsis”, merging *Personal Information Management* with Semantic Web technologies. After this, I continued working on this topic at the German Research Center for Artificial Intelligence DFKI

since 2004 in the NEPOMUK project. My research focus is on Semantic Web and its use in Knowledge Management. In practice I contribute to open source projects, am member of the W3C SWEO Interest Group, and present these results to the public. Privately, I am a believing Christian and enjoy netculture and the digital revolution.

My personal homepage is `http://leobard.net` where I also keep my foaf file, `http://www.leobard.net/rdf/foaf.xml`.

Bibliography

[ABM+00] Andreas Abecker, Ansgar Bernardi, Heiko Maus, Michael Sintek, and Claudia Wenzel. Information supply for business processes: coupling workflow with document analysis and information retrieval. *Knowledge Based Systems*, 13(5):271–284, 2000.

[AG] Brainbot AG. the brainfiler text classification system. `http://www.brainbot.de`.

[AKS99] E. Adar, D. Karger, and L. Stein. Haystack: Per-user information environments. In *Proceedings of the 1999 Conference on Information and Knowledge Management, CIKM*, 1999.

[Ape05] Aperture. http://aperture.sourceforge.net, 2005.

[ARD06] Stefan Agne, Christian Reuschling, and Andreas Dengel. Dynaq - dynamic queries for electronic document management. In Thomas Kwok and William Cheung, editors, *Proceedings of the First International Electronic Document Management in an Enterprise Computing Environment Workshop (EDM 2006) at the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, pages pp. 56–59, Hong Kong, China, 17 October 2006. IEEE.

[ASRB07] Benjamin Adrian, Leo Sauermann, and Thomas Roth-Berghofer. ConTag: A semantic tag recommendation system. In Tassilo Pellegrini and Sebastian Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 297–304. JUCS, 2007.

[Ass98] Association for Computing Machinery, Inc. The ACM Computing Classification System. `http://www.acm.org/class/`, 1998. Valid 2007.

[AvE04] Andreas Abecker and Ludger van Elst. *Handbook on Ontologies*, chapter Ontologies for Knowledge Management, pages pp. 435–454, Springer. Springer, 2004.

[Bar95] Deborah K. Barreau. Context as a factor in personal information management systems. *J. Am. Soc. Inf. Sci.*, 46(5):327–339, 1995.

[BB05] Mary Bazire and Patrick Brézillon. Understanding context before using it. In Anind K. Dey, Boicho N. Kokinov, David B. Leake, and Roy M. Turner, editors, *CONTEXT*, volume 3554 of *Lecture Notes in Computer Science*, pages 29–40. Springer, 2005.

[BC06] Christian Bizer and Richard Cyganiak. D2R-Server - Publishing Relational Databases on the Web as SPARQL-Endpoints. In *Proc. of the 15th International World Wide Web Conference (WWW2006)*, 2006.

[Bec06] Dave Beckett. Semantics Through the Tag. XTech presentation `http://xtech06.usefulinc.com/schedule/paper/135`, 2006.

[BG04] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. `http://www.w3.org/TR/rdf-schema/`, 10 February 2004.

[BGSV06] Stephan Bloehdorn, Olaf Görlitz, Simon Schenk, and Max Völkel. TagFS - Tag Semantics for Hierarchical File Systems. In *Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06), Graz, Austria, September 6-8, 2006*, SEP 2006.

[BL] Tim Berners-Lee. Frequently asked questions by the press. `http://www.w3.org/People/Berners-Lee/FAQ.html`. Last visited December 2007.

[BL98] Tim Berners-Lee. Notation 3. Design issue, W3C, http://www.w3.org/DesignIssues/Notation3.html, 1998.

[BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter. RFC3986: Uniform Resource Identifier (URI): Generic Syntax. `http://www.ietf.org/rfc/rfc3986.txt`, January 2005.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 89, May 2001.

[BLP80] Tim Berners-Lee and Sean B Palmer. Enquire manual - in hypertext. Historical W3C Document, October 1980. This is a HyperText rendition of Tim Berners-Lee's original ENQUIRE V 1.1 manual from October 1980, painstakingly copied by Sean B. Palmer.

[BN95] Deborah Barreau and Bonnie A Nardi. Finding and reminding: File organization from the desktop. 1995.

[Boa04] Richard Boardman. *Improving Tool Support for Personal InformationManagement*. PhD thesis, Department of Electrical and Electronic Engineering Imperial College London University of London, July 13 2004.

[BPM05] Paul Buitelaar, PhilippCimiano, and Bernardo Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, July 2005.

[Bru98] Roland Bruenken. *Automatische Rekonstruktion von Inhaltsbeziehungen zwischen Dokumenten - Benutzeradaptiver Zugriff auf Wissensbasen*. PhD thesis, Aachen, 1998.

[BS04] Richard Boardman and M. Angela Sasse. "stuff goes into the computer and doesn't come out": a cross-tool study of personal information management. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 583–590, New York, NY, USA, 2004. ACM.

[BTMC04] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373, 2004.

[Bus45] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1):p101–108, July 1945.

[CAL06] K. Votis C. Alexakos, B. Vassiliadis and S. Likothanassis. *A Multilayer Ontology Scheme for Integrated Searching in Distributed Hypermedia*, volume Adaptive and Personalized Semantic Web of *Studies in Computational Intelligence*. Springer, 2006.

[Car00] John M. Carroll. *Making Use: Scenario-Based Design of Human-Computer Interactions*. MIT Press, Cambridge, MA, USA, 2000.

[Cay04] Steve Cayzer. Semantic blogging and decentralized knowledge management. *Commun. ACM*, 47(12):47–52, 2004.

[CB04] A. Seaborne C. Bizer. D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

[CBHS05] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM Press.

[CDK+07] T. Catarci, A. Dix, A. Katifori, G. Lepouras, and A. Poggi. Task-centered information management. In C. Thanos and F. Borri, editors, *DELOS Conference 2007 on Working Notes*, pages 253–263, Tirrenia, Pisa, 13-14 February 2007.

[CGG+04] Paul-Alexandru Chirita, Rita Gavriloaie, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Activity based metadata for semantic desktop search. 2004.

[Cla96] H. H. Clark. Using language. *Cambridge University Press*, 1996.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, second edition edition, 2001.

[CMBT02] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[CoHF87] National Research Council Committee on Human Factors. *Mental Models in Human-Computer Interaction: Research Issues About What the User of Software Knows*. The National Academic Press, 1987.

[CPG05] Adam Cheyer, Jack Park, and Richard Giuli. IRIS: Integrate. Relate. Infer. Share. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.

[CR03] Alan Cooper and Robert M. Reimann. *About Face 2.0: The Essentials of Interaction Design*. Wiley Publishing, Inc., Indianapolis, Indiana, 2003.

[Cra43] K. Craik. *The Nature of Explanation*. Cambridge University Press, London, New York., (1943).

[DAB+02] Andreas Dengel, Andreas Abecker, Jan-Thies Bähr, Ansgar Bernardi, Peter Dannenmann, Ludger van Elst, Stefan Klink, Heiko Maus, Sven Schwarz, and Michael Sintek. EPOS – Evolving Personal to Organizational Knowledge Spaces, 2002.

[Den06a] Andreas Dengel. Vorlesung wissensmanagement. Lecture TU Kaiserslautern, 2006.

[Den06b] Andreas R. Dengel. Six Thousand Words about Multi-Perspective Personal Document Management. In *Proc. EDM IEEE Workshop*. IEEE, Oct 2006.

[DF04] Stefan Decker and Martin Frank. The social semantic desktop. In *Proc. of the WWW2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web*, 2004.

[DH05] Xin Dong and Alon Y. Halevy. A platform for personal information management and integration. In *Proc. of the CIDR Conference*, pages 119–130, 2005.

[Dou98] M. Dougiamas. A journey into constructivism. website, http://dougiamas.com/writing/constructivism.html, November 1998.

[DPQS05] Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors. *The Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure. Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland*, volume 175 of *CEUR Workshop Proceedings ISSN 1613-0073*, November 2005.

[DPS+06] Stefan Decker, Jack Park, Leo Sauermann, Sören Auer, and Siegfried Handschuh, editors. *Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop (SemDesk 2006) located at the 5th International Semantic Web Conference ISWC 2006*, volume 202 of *CEUR-WS*, Athens, GA, USA, 6th November 2006 2006.

[EAD+06] Katja Einsfeld, Stefan Agne, Matthias Deller, Achim Ebert, Bertin Klein, and Christian Reuschling. Dynamic visualization and navigation of semantic virtual environments. In Ebad Banissi, Remo Aslak Burkhard, Anna Ursyn, Jian J. Zhang, Mark Bannatyne, Carsten Maple, Andrew J. Cowell, Gui Yun Tian, and Ming Hou, editors, *Proceedings of the 10th International Conference on Information Visualization (IV 2006)*, pages 569–574, London, United Kingdom, 5-7 July 2006. IEEE Computer Society.

[(ed04] Alistair Miles (edt). Simple Knowledge Organisation System (SKOS). `http://www.w3.org/2004/02/skos/`, Feb 2004.

[FG96] Eric Freeman and David Gelernter. Lifestreams: A storage model for personal data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(1):pp80, 1996.

[FGSSB06] Norberto Fernandez-Garcia, Leo Sauermann, Luis Sanchez, and Ansgar Bernardi. PIMO Population and Semantic Annotation for the Gnowsis Semantic Desktop. In *Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop at the ISWC*, volume 202 of *CEUR-WS*, 2006.

[FM04] Editors F. Manola, E. Miller. RDF Primer. W3C Recommendation, W3C, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, 10 February 2004.

[FS03] John Feather and Paul Sturges. International encyclopedia of information and library science. *Inf. Res.*, 8(3), 2003.

[Gal05] David Galbraith. Tagging and the Semantic Web. `http://www.davidgalbraith.org/archives/000764.html`, April 2005. Blogpost.

[GB08] Saul Greenberg and Bill Buxton. Usability evaluation considered harmful (some of the time). In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 111–120, New York, NY, USA, 2008. ACM.

[GBL+02] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. Mylifebits: Fulfilling the memex vision. In *ACM Multimedia December 1-6, Juan-les-Pins, France*, pages pp. 235–238, 2002.

[Gel05] Joe Geldart. RDF without Revolution An Analysis and Test of RDF and Ontology. Bachelor thesis, Department of Computer Science, University of Durham, April 2005.

[GHM+07] Tudor Groza, Siegfried Handschuh, Knud Moeller, Gunnar Grimnes, Leo Sauermann, Enrico Minack, Cedric Mesnage, Mehdi Jazayeri, Gerald Reif, and Rosa Gudjonsdottir. The NEPOMUK Project — On the way to the Social Semantic Desktop. In Tassilo Pellegrini and Sebastian Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 201–211. JUCS, 2007.

[GKV+06] M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis. User profile ontology version 1. http://oceanis.mm.di.uoa.gr/pened/?category=publications, 2006.

[Gla90] E Von Glasersfeld. *Constructivist views on the teaching and learning of mathematics*, chapter An exposition of constructivism: Why some like it radical., pages 19–29. Virginia: National Council of Teachers of Mathematics, 1990.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GS83] D. Gentner and A.L. Stevens. *Mental Models*. Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1983.

[Har04] Andreas Harth. SECO: mediation services for semantic web data. *Intelligent Systems, IEEE*, Volume 19(Issue 3):66 – 71, May-June 2004.

[Hei06] Dominik Heim. Semantic Wikis in knowledge management - Evaluating the Gnowsis approach. Master's thesis, Fachhochschule Kaiserslautern, August 2006.

[HMBR05] Harald Holz, Heiko Maus, Ansgar Bernardi, and Oleg Rostanin. From Lightweight, Proactive Information Delivery to Business Process-Oriented Knowledge Management. volume 0, pages 101–127, 2005.

[Hor06] Benjamin Horak. ConTag - A Tagging System linking the Semantic Desktop with Web 2.0. Diploma thesis, University Kaiserslautern, http://www.dfki.uni-kl.de/ horak/mypubs/ConTag.pdf, August 2006.

[HT97] Hardy and Taylor. Von Glasersfeld's Radical Constructivism: A Critical Review. *Science and Education (Kluwer)*, 6:pp 135–150, 1997.

[HvOH06] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A Browser for Heterogeneous Semantic Web Repositories. In Cruz et al., editor, *Proc. of the International Semantic Web Conference ISWC2006*, pages pp. 272–285, Athens, USA, November 2006.

[IAD06] Jon Iturrioz, Sergio F. Anzuola, and Oscar Díaz. Turning the mouse into a semantic device: the seMouse experience. In York Sure and John Domingue, editors, *Proc. of the 3rd European Semantic Web Conference, ESWC 2006 Budva, Montenegro, June 11-14, 2006*, volume 4011 of *Springer LNCS*, 2006.

[INN08] Ekaterini Ioannou, Claudia NiederÃ©e, and Wolfgang Nejdl. Probabilistic entity linkage for heterogeneous information spaces. *Advanced Information Systems Engineering*, 5074:556–570, 2008.

[IV05] Indratmo and Julita Vassileva. Human and social aspects of decentralized knowledge communities. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.

[JB05] William Jones and Harry Bruce. A Report on the NSF-Sponsored Workshop on Personal Information Management, Seattle, WA. Technical Report, January 2005. http://pim.ischool.washington.edu/report2005

[JL83] P. N. Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness.* Cambridge, MA: Harvard University Press, 1983.

[JMBF05] William Jones, Charles F. Munat, Harry Bruce, and Austin Foxley. The Universal Labeler: Plan the Project and Let Your Information Follow. In Grove and Andrew, editors, *Proceedings 68th Annual Meeting of the American Society for Information Science and Technology (ASIST)*, 2005.

[Jon08] William Jones. *Keeping Found Things Found.* Morgan Kaufmann Publishers, Elsevier, 2008.

[JT07] William Jones and Jaime Teevan, editors. *Personal Information Management*. University of Washington Press, October 2007.

[Kel06] Diane Kelly. Evaluating personal information management behaviors and tools. *Communications of the ACM*, 49(1):84–86, January 2006.

[Kie06] Malte Kiesel. Kaukolu: Hub of the semantic corporate intranet. In *Proc. of the Workshop SemWiki2006 - From Wiki to Semantics at the ESWC Conference*, 2006.

[KJ06] David R. Karger and William Jones. Data unification in personal information management. *Commun. ACM*, 49(1):77–82, 2006.

[Koy01] I. Koychev. Learning about user in the presence of hidden context. In *In Proceedings of Machine Learning for User Modeling: UM-2001.*, 2001.

[KS05] Malte Kiesel and Leo Sauermann. Towards semantic desktop wikis. *UPGRADE special issue on "The Semantic Web"*, VI:30 – 34, 2005.

[Kun03] Mike Kuniavsky. *Observing the User Experience*. Morgan Kaufmann Publishers, 2003.

[KVV05] M. Krötzsch, D. Vrandecic, and M. Völkel. Wikipedia and the semantic web - the missing links, 2005.

[Lee00] Tim Berners Lee. *Weaving the Web, The Past, Present and Future of the World Wide Web by its Inventor*. Texere, London, 2000.

[LR94] Clayton Lewis and John Rieman. *Task-centered user interface design: A practical introduction*. 1994.

[LT06] Khalid Latif and A Min Tjoa. Combining context ontology and landmarks for personal information management. In *Proceedings of International Conference on Computing and Informatics (ICOCI)*, Kuala Lumpur, Malaysia, June 2006.

[MA05] Tristan Miller and Stefan Agne. Attention-based information retrieval using eye tracker data. In *Proceedings of the Third International Conference on Knowledge Capture (K-CAP05)*, September 2005. To appear.

[MAD05] Tristan Miller, Stefan Agne, and Andreas Dengel. eFISK – eine aufmerksamkeitsbasierte Schlüsselwort-Extraktions- und Information Retrieval-Maschine. Abschlussbericht 15202-386261/659, Stiftung Rheinland-Pfalz für Innovation, June 2005.

[Man02] Farhad Manjoo. Blah, blah, blah and blog. http://www.wired.com/culture/lifestyle/news/2002/02/50443, February 2002. accessed 20.10.2007.

[McB01] Brian McBride. Jena: Implementing the rdf model and syntax specification. In *Proc. of the Semantic Web Workshop WWW2001*, 2001.

[MEHL04] L. McDowell, O. Etzioni, A. Halevey, and H. Levy. Semantic email, 2004.

[MH04] D. L. McGuinness and F. Harmelen. OWL Web Ontology Language Overview W3C Recommendation. http://www.w3.org/TR/owl-features/, 10 February 2004.

[MHBR05] Heiko Maus, Harald Holz, Ansgar Bernardi, and Oleg Rostanin. Leveraging passive paper piles to active objects in personal knowledge spaces. In *Proceedings of 3rd Conference Professional Knowledge Management: Experiences and Visions*, pages 43–46, April 2005.

[MMY04] Robert MacGregor, Sameer Maggon, and Baoshi Yan. MetaDesk: A Semantic Web Desktop Manager. In *In Knowledge Markup and Semantic Annotation Workshop, ISWC 2004*, November 2004.

[Moo06] David S. Moore. *The Basic Practice of Statistics*. W. H. Freeman, 4th edition, 2006.

[Mor02] Emile L. Morse. Evaluation methodologies for information management systems. *D-Lib Magazine*, 8(9), September 2002.

[MS06] Enrico Minack and Leo Sauermann. Adapters, Extractors, and Knowledge Structure Services Deliverable D2.1. EU Project Deliverable D 2.1, NEPOMUK Consortium, Dec 2006. Contributors: Stefania Costache, Julien Gaugaz, Gunnar Grimnes, Antoni Myłka, Max Völkel.

[MSS01] Alexander Maedche, Steffen Staab, and Rudi Studer. Ontologien. *Wirtschaftsinformatik*, 43(4):393–396, 2001.

[Nel72] Ted Nelson. As we will think. *On-line 72 Conference Proceedings*, vol. 1:pp. 439–454, 1972.

[Nie00] Jacob Nielsen. Why you only need to test five users. http://www.useit.com/alertbox/20000319.html, March 2000.

[OKS05] Frank Osterfeld, Malte Kiesel, and Sven Schwarz. Nabu - A Semantic Archive for XMPP Instant Messaging. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.

[OR23] Charles Kay Ogden and Ivor Richards. *The meaning of meaning.- A study of the influence of language upon thought and of the science of symbolism.* London: Kegan Paul, Trench, Trubner & Co., 1923. With an introduction by J.P. Postgate and supplementary essays by B. Malinowski and F. G. Crookshank.

[Ore05] Eyal Oren. SemperWiki: a semantic personal Wiki. In *Proceedings of the 1st Semantic Desktop Workshop at the ISWC2005*, 2005.

[Pe05] Eric Prud'hommeaux and Andy Seaborne (edts). Sparql query language for rdf. W3c working draft, W3C, 2005. http://www.w3.org/TR/rdf-sparql-query/.

[QHK03] Dennis Quan, David Huynh, and David R. Karger. Haystack: A platform for authoring end user semantic web applications. In *International Semantic Web Conference*, pages 738–753, 2003.

[Qua03] Dennis Quan. *Designing End User Information Environments Built on Semistructured Data Models*. PhD thesis, MIT, 2003.

[Rat03] Holger Rath. The Topic Maps Handbook Detailed description of the standard and practical guidelines for using it in knowledge management. empolis white paper, empolis GmbH, 2003.

[RBB+02] Janice Redish, Randolph G. Bias, Robert Bailey, Rolf Molich, Joe Dumas, and Jared M. Spool. Usability in practice: formative usability evaluations - evolution and revolution. In *CHI '02 extended abstracts on Human factors in computing systems*, pages 885–890, 2002.

[RCL+98] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data mountain: using spatial memory for document management. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 153–162, New York, NY, USA, 1998. ACM Press.

[Rij79] C. J. van Rijsbergen. *Information retrieval*. Butterworths, London, 2 edition, 1979.

[RMG06] Gerald Reif, Martin Morger, and Harald C. Gall. Semantic Clipboard - Semantically Enriched Data Exchange Between Desktop Applications. In *Semantic Desktop and Social Semantic Collaboration Workshopat the 5th International Semantic Web Conference ISWC06*, Athens, Geogria, USA, November 2006.

[Roh05] Jean Rohmer. Lessons for the future of Semantic Desktops learnt from 10 years of experience with the IDELIANCE Semantic Networks Manager. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.

[Row94] David E. Rowley. Usability testing in the field: bringing the laboratory to the user. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, pages 252–257, 1994.

[SAP05] SAP AG. CRM on demand system. Video-Presentation, visited March 2005. http://www.sap.com/solutions/business-suite/crm/crmondemand/index.epx.

[Sau03] Leo Sauermann. The Gnowsis—Using Semantic Web Technologies to build a Semantic Desktop. Diploma thesis, Technical University of Vienna, 2003.

[Sau05] Leo Sauermann. The Semantic Desktop—a Basis for Personal Knowledge Management. In Hermann Maurer, Cristian Calude, Arto Salomaa, and Klaus Tochtermann, editors, *Proceedings of the I-KNOW 2005. 5th International Conference on Knowledge Management*, pages 294 – 301. IICM/J.UCS, 2005.

[Sau06] Leo Sauermann. PIMO—a PIM Ontology for the Semantic Desktop (draft). Draft, DFKI, 2006.

[SBD05] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and Outlook on the Semantic Desktop. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, volume 175 of *CEUR Workshop Proceedings*, pages pp. 1–19. CEUR-WS, November 2005. http:// CEUR-WS.org/Vol-175/.

[Sch05] Sven Schwarz. A context model for personal knowledge management. In *Proceedings of the IJCAII'05 Workshop on Modeling and Retrieval of Context*, Edinburgh, 2005.

[Sch06] Sven Schwarz. A context model for personal knowledge management applications. In Thomas Roth-Berghofer, Stefan Schulz, and David B. Leake, editors, *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK, July 31 - August 1, 2005, Revised Selected Papers*, volume 3946 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006.

[SCV07] Leo Sauermann, Richard Cyganiak, and Max Völkel. Cool URIs for the Semantic Web. Technical Memo TM-07-01, DFKI GmbH, Deutsches Forschungszentrum f ür Künstliche Intelligenz GmbH Postfach 2080 67608 Kaiserslautern, February 2007. Written by 29.11.2006.

[SDvE+06] Leo Sauermann, Andreas Dengel, Ludger van Elst, Andreas Lauer, Heiko Maus, and Sven Schwarz. Personalization in the EPOS project. In *Proceedings of the Semantic Web Personalization Workshop at the ESWC 2006 Conference*, pages 42 – 52, 2006.

[SEM07] Leo Sauermann, Ludger Van Elst, and Knud Möller. Personal Information Model (PIMO) Ontology Guide. NEPOMUK Specification, 12 2007. http://www.semanticdesktop.org/ontologies/pimo.

[SGK+06] Leo Sauermann, Gunnar Aastrand Grimnes, Malte Kiesel, Christiaan Fluit, Heiko Maus, Dominik Heim, Danish Nadeem, Benjamin Horak, and Andreas Dengel. Semantic Desktop 2.0: The Gnowsis Experience. In *Proc. of the ISWC Conference*, volume 4273/2006 of *Lecture Notes in Computer Science*, pages 887–900. Springer Berlin / Heidelberg, Nov 2006. ISSN 0302-9743 (Print) 1611-3349 (Online).

[SH08] Leo Sauermann and Dominik Heim. Evaluating long-term use of the gnowsis semantic desktop for pim. In *Proc. ISWC Conference*, volume 5318 of *LNCS*, pages pp 467–482, 2008.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:pp. 379–423, July 1948.

[Sie05] Mark Siebert. Knowledge creation framework - enabling just-in-time information delivery. In Klaus-Dieter Althoff, Andreas Dengel, Ralph Bergmann, Markus Nick, and Thomas Roth-Berghofer, editors, *Wissensmanagement (LNCS Volume)*, volume 3782 of *Lecture Notes in Computer Science*, pages 699–709. Springer, 2005.

[SP01] Graham Moore (edss). Steve Pepper. XML Topic Maps (XTM) 1.0. Specification, TopicMaps.Org, 2001.

[SRB03] Sven Schwarz and Thomas Roth-Berghofer. Towards goal elicitation by user observation. In *Proceedings of the FGWM 2003 Workshop on Knowledge and Experience Management*, Karlsruhe, 2003.

[SS05a] Leo Sauermann and Sven Schwarz. Gnowsis adapter framework: Treating structured data sources as virtual rdf graphs. In *Proceedings of the ISWC 2005*, 2005.

[SS05b] Roza Shkundina and Sven Schwarz. A similarity measure for task contexts. In *Proceedings of the Workshop Similarities - Processes - Workflows in conjunction with the 6th International Conference on Case-Based Reasoning (ICCBR 2005)*, Chicago, aug 2005.

[SSSD06] Mark Siebert, Pierre Smits, Leo Sauermann, and Andreas Dengel. Increasing search quality with the semantic desktop in proposal development. In *Proceedings of the Practical Aspects of Knowledge Management PAKM conference*, volume 4333/2006 of *Lecture Notes in Computer Science*, pages 279–290. Springer Berlin / Heidelberg, 2006.

[Sve00] Elaine Svenonius. *The intellectual foundation of information organization*. MIT Press, Cambridge, MA, USA, 2000.

[SvED07] Leo Sauermann, Ludger van Elst, and Andreas Dengel. PIMO - a Framework for Representing Personal Information Models. In Tassilo Pellegrini and Sebastian Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 270–277. JUCS, 2007.

[TAAK04] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422, New York, NY, USA, 2004. ACM.

[THD08] VinhTuan Thai, Siegfried Handschuh, and Stefan Decker. Tight coupling of personal interests with multi-dimensional visualization for exploration and analysis of text collections. In *International Conference on Information Visualisation*, volume 0, pages 221–226, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[TJB06] Jaime Teevan, William Jones, and Benjamin B. Bederson. Introduction. *Commun. ACM*, 49(1):40–43, 2006.

[TMN06] Giovanni Tummarello, Christian Morbidoni, and Michele Nucci. Enabling Semantic Web Communities with DBin: An Overview. In *Proceedings of the ISWC*, pages 943–950, 2006.

[TMPP05] G. Tummarello, C. Morbidoni, P. Puliti, and F. Piazza. The DBin Semantic Web platform: an overview. In *Proc. of the WWW2005 Workshop on The Semantic Computing Initiative SeC*, 2005.

[vEK04] Ludger van Elst and Malte Kiesel. Generating and integrating evidence for ontology mappings. In *Engineering Knowledge in the Age of the Semantic Web: Proceedings of the 14th International Conference, EKAW 2004*, volume 3257 of *LNAI*, pages 15–29, Heidelberg, 2004. Springer.

[WSLW01] Richard Saul Wurman, David Sume, Loring Liefer, and Karen Whitehouse. *Information Anxiety 2*. Que, 2001. isbn:978-0789724106.

[WW08] Wolfgang Woerndl and Maximilian Woehrl. SeMoDesk: Towards a Mobile Semantic Desktop. In *Proc. PIM Workshop, CHI*, 2008.

[XC05] Huiyong Xiao and Isabel F. Cruz. A multi-ontology approach for personal information management. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.

