ISWC2005
International Semantic Web Conference
November 6-10 2005
Galway, Ireland
http://iswc2005.semanticweb.org

W2: The Semantic Desktop—Next Generation Information Management & Collaboration Infrastructure

Organisers: Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann

Sunday, 6 November 2005

ISWC 2005 could not take place without the generous support of the following sponsors

Super Emerald Sponsors

Gold Sponsors

Silver Sponsors

ISWC 2005 Organising Committee

General Chair	Mark Musen, Stanford University
Research Track Co-Chair	Yolanda Gil, Information Sciences Institute
Research Track Co-Chair	Enrico Motta, The Open University
Industrial Track Chair	V Richard Benjamins, iSOCO, S.A.
Workshop Chair	Natasha F Noy, Stanford University
Tutorial Chair	R.V. Guha, Google
Poster & Demo Chair	Riichiro Mizoguchi, Osaka University
Semantic Web Challenge	Michel Klein, Vrjie Universiteit Amerdam
Semantic Web Challenge	Ubbo Visser, Universitat Bremen
Doctoral Symposium Co-Chair	Edward Curry, National University of Ireland, Galway
Doctoral Symposium Co-Chair	Enda Ridge, University of York
Meta-Data Chair	Eric Miller, W3C
Sponsorship Chair	Liam O'Móráin, DERI Galway
Local Organising Co-Chair	Christoph Bussler, DERI Galway
Local Organising Co-Chair	Stefan Decker, DERI Galway
Local Organiser	Brian Cummins, DERI Galway
Webmaster	Seaghan Moriarty, DERI Galway
Web Design	Johannes Breitfuss, DERI Innsbruck

Preface

The Internet, electronic mail, and the Web have revolutionized the way we communicate and collaborate - their mass adoption is one of the major technological success stories of the 20th century. We all are now much more connected, and in turn face new resulting problems: information overload caused by insufficient support for information organization and collaboration. For example, sending a single file to a mailing list multiplies the cognitive processing effort of filtering and organizing this file times the number of recipients—leading to more and more of peoples' time going into information filtering and information management activities. There is a need for smarter and more fine-grained computer support for personal and networked information that has to blend the boundaries between personal and group data, while simultaneously safeguarding privacy and establishing and deploying trust among collaborators.

The Semantic Web holds promises for information organization and selective access, providing standard means for formulating and distributing metadata and Ontologies.

Still, we miss a wide use of Semantic Web technologies on personal computers. The use of ontologies, metadata annotations, and semantic web protocols on desktop computers will allow the integration of desktop applications and the web, enabling a much more focused and integrated personal information management as well as focused information distribution and collaboration on the Web beyond sending emails. The vision of the Semantic Desktop for personal information management and collaboration has been around for a long time: visionaries like Vanevar Bush and Doug Engelbart have formulated and partially realized these ideas. However, for the largest part their ideas remained a vision for far too long since the foundational technologies necessary to render their ideas into reality were not yet invented—these ideas were proposing jet planes, where the rest of the world had just invented the parts to build a bicycle. However, recently the computer science community has developed the means to make this vision a reality:

- The Semantic Web effort[1] provides standards and technologies for the definition and exchange of metadata and ontologies.
- Open-source software (such as OpenOffice) make it possible to reuse and build on top of existing sophisticated systems.
- Collaboration, acquisition and dissemination infrastructures such as Wikis and Blogs are providing the foundation for joint collaborative knowledge creation
- Social Software maps the social connections between different people into the technical infrastructure.

[1] http://www.w3.org/sw

– P2P and Grid computing, especially in combination with the Semantic Web field, develops technology to interconnect large communities.

The application of the mentioned technologies, especially in combination with the Semantic Web, to the desktop computer in order to improve personal information management and collaboration was the main topic of this workshop. Several systems have been created already to explore this field, e.g., the Haystack system at MIT, the Gnowsis system at DFKI, or the Chandler system by the OSA foundation. The main focus of this workshop was on providing an overview of existing approaches and elaborating the next steps necessary in order to bring the Semantic Web to the desktop computer.

These workshop proceedings contain the papers submitted to and presented at the workshop. The high number of submissions reflected substantial interest and activity in this area. Special thanks go to the program committee and reviewers, for their efforts and hard work in the reviewing and selection process.

Kaiserslautern, October 2005

Stefan Decker
Jack Park
Dennis Quan
Leo Sauermann

Workshop Chairs

Stefan Decker (DERI, National University of Ireland, Galway)
Jack Park (SRI International, Menlo Park, USA)
Dennis Quan (IBM, T.J. Watson Research Center, USA)
Leo Sauermann (DFKI, Kaiserslautern, Germany)

Program Comittee

Andreas Abecker (FZI, Karlsruhe, Germany)
Dan Brickley (W3C, Sophia Antipolis, France)
Jeen Broekstra (Aduna BV, Netherlands)
Pat Croke (Hewlett Packard, Galway, Ireland)
Isabel Cruz (University of Illinois at Chicago, USA)
Hamish Cunningham (University of Sheffield, UK)
Doug Engelbart (Bootstrap Institute, USA)
Dirk-Willem van Gulik (Apache Foundation, Netherlands)
Manfred Hauswirth (EPFL, Switzerland)
Malte Kiesel (DFKI, Germany)
Stéphane Laurière (Mandriva, France)
Gregoris Mentzas (National Technical University of Athens, Greece)
Peter Mika (Vrije Universiteit Amsterdam)
Wolfgang Nejdl (L3S, Hannover)
David O'Sullivan (DERI, NUIG, Ireland)
Simon Phipps (Sun, UK, USA)
Wolfgang Prinz (Fraunhofer and RWTH Aachen, Germany)
Gerald Reif (TU Vienna, Austria)
Thomas R. Roth-Berghofer (TU Kaiserslautern/DFKI, Germany)
Daniel Schwabe (PUC Rio, Brasil)
David Schwartz (Bar Ilan University, Israel)
Michael Sintek (DFKI, Germany)
Heiner Stuckenschmidt (Free University of Amsterdam, The Netherlands)
Giovanni Tummarello (Universita' Politecnica delle Marche, Italy)

Table of Contents

Posters

Overview and Outlook on the Semantic Desktop

Leo Sauermann, Ansgar Bernardi, Andreas Dengel

Knowledge Management Department
German Research Center for Artificial Intelligence DFKI GmbH,
Erwin-Schrödinger-Straße 57, 67663 Kaiserslautern, Germany
{leo.sauermann|ansgar.bernardi|andreas.dengel}@dfki.de

Abstract. In this paper we will give an overview of the Semantic Desktop paradigm, beginning with the history of the term, a definition, current work and its relevance to knowledge management of the future. Existing applications and research results are listed and their role as building blocks of the future Semantic Desktop described. Based on the analysis of existing systems we propose two software architecture paradigms, one for the Semantic Desktop at large and another for applications running on a Semantic Desktop. A view on the context aspect of the Semantic Desktop and the Knowledge Management aspect is given. Based on the current events and projects, we give an outlook on the next steps.

1 Introduction

The Semantic Desktop will be the driving paradigm for desktop computing in the area of the Semantic Web. Based on the needs and expectations of users today the software industry will evolve to a future way of computing, semantic desktop computing.

The main task at hand is to **transfer the Semantic Web to desktop computers**, and this transfer will not only consist of the technology, but also of the philosophy and the people involved. Developers that today concentrate on services for the Semantic Web (and find tools and examples) will need a complete RDF and ontology based environment to create applications on desktop computers. End users will benefit from these applications, as they integrate and also communicate better—based on ontologies and Semantic Web standards—than today's desktop applications.

1.1 The background and goals of the Semantic Desktop community

In 1945, Vannevar Bush wrote the now famous article "As we may think" [1], where he described the visionary system called "Memex". The definition that he gave was important for many systems to follow:

> 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 based his ideas solely on analog devices, running on punch cards and using microfilm as storage. Today we notice how his vision becomes reality, the personal computer is very close to what Bush had in mind. Not all books and records are stored in a PC, but we are close to it. The idea of *trails*—paths of resources that build a personal look on a topic—were taken up by system like lifestreams [2]. 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*" [3]. 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 the Memex were built, it never ignited the revolution that was intended by Nelson. In 1992 the World Wide Web launched, created by Tim Berners-Lee. The Web grew at a very fast rate and changed society; information is used in a different way than in the pre-web era. Before the web lifted off, Berners-Lee programmed the *Enquire-Within-Upon-Everything* system. *Enquire* was a personal information management tool to store information about people, projects, hardware resources and how they relate to each other. It was 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.[4, p. 17]

These were the requirements that led to a distributed version of Enquire that we know today as the World Wide Web [5]. 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" is still a vivid arena with many competing companies. The problem of metadata and labeled links was identified and is now tackled by the Semantic Web Initiative [6].

1.2 Today's state of the Semantic Desktop idea

In 2003, facing the fact that the Semantic Web was not universally used, we analyzed the field and found that the major projects aimed at large and distributed organizations, but the end user was only supported by Haystack or Protege, which both were complicated. Nearly all information we saw on web pages and in electronic documents had been created by people using personal computers. The PC was the place where most personal data is stored and the major interface to the web. Information stored on a server was usually manipulated through interfaces that are executed on a PC, be it a web browser or web authoring tool. The use of ontologies, classifications and global identifiers in normal desktop applications did not happen. Tim Berners-Lee also realized that the

end user applications were missing and requested in several talks that we start building useful applications. From this perspective we stated [7]:**If the goal is to have a global Semantic Web, one building block is a Semantic Desktop, a Web for a single user.**

The term "Semantic Desktop" itself was coined by Stefan Decker and picked up by Leo Sauermann in 2003, to create a term that creates a mutual understanding for the similar ideas. Stefan Decker and Martin Frank stated the need for a "Networked Semantic Desktop" [8] in 2004 and sketched the way to the events today. 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"*. His roadmap to the *Networked Semantic Desktop* is laid out as follows: [8]

- 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 [8, 7, 9] we could define a Semantic Desktop in the following way:

Definition 1. *A Semantic Desktop is a device in which an individual stores all her digital information like documents, multimedia and messages. These are interpreted as Semantic Web resources, each is identified by a Uniform Resource Identifier (URI) and all data is accessible and queryable as RDF graph. Resources from the web can be stored and authored content can be shared with others. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems. Applications respect this and store, read and communicate via ontologies and Semantic Web protocols. The Semantic Desktop is an enlarged supplement to the user's memory.*

1.3 The near future

From our point of view, we have achieved most of the goals of the first phase and are currently in the second phase. Our task is now to weave the existing and stable parts of the Semantic Web into desktop computing, P2P, and Social Networking. In this paper we will address the aspects of a single Semantic Desktop system, the role of a Semantic Desktop in a networked environment was already addressed by Decker et al. [8]. The Semantic Desktop is a global project involving researchers and industry from different technical fields.

To create a focal point for the Semantic Desktop, the European IST Project NEPOMUK was initiated by a consortium lead by the DFKI. Bringing together researcher 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, this project will build 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*. It enables users to build, maintain, and employ inter-workspace relations in large scale distributed scenarios. New knowledge can be articulated in semantic structures and be connected with existing information items on the local and remote desktops. Knowledge, information items, and their metadata can be shared spontaneously without a central infrastructure. NEPOMUK will realize a freely available open-source integration framework with a set of standardized interfaces, ontologies and applications. Collaboration with the open source community and integration with major open source products is intended and will ensure the broad acceptance of NEPOMUK technology—thereby activating a sustainable open source movement with viral spread-out. 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 [4]) – the Semantic Desktop community at large will gain momentum through this project.

2 Semantic Desktop building blocks

To provide such a system to end users, a few prerequisites are required. In this section we start describing research projects that address the topic of an integrated Semantic Desktop and then we give examples of tools that are available today as building blocks for the future Semantic Desktop. An outlook will be given on the features users can expect and the relevance to personal knowledge management.

2.1 Integrated projects

The first research project using the term was the *Gnowsis Semantic Desktop* [7] by Leo Sauermann, co-author of this paper. The work was a diploma thesis and deals with the details of integrating desktop data sources into a unified RDF graph, also addressing the problem of how to identify resources with URIs. You will find an introduction to the field in this work and a prototypical user interface, introducing the terms "link and browse" as a desktop metaphor. The project is now continued both as an open-source project and is reused framework for other research projects, namely *EPOS* [10] and *@Visor*.

Similar to the gnowsis work, but on the web-services world is the *SECO: mediation services for semantic Web data* project aiming at integrating web sources [11]. It

describes an infrastructure that lets agents uniformly access data that is potentially scattered across the Web. The results can be transformed to the desktop, as we have done in another paper submitted to ISWC [12]. In the field of data integration, also the architecture by Bizer and Seaborne [13] about adapting SQL sources should be mentioned. A product by the Microsoft corporation called *Information Bridge Framework* [14] 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 applications, that are extended by plugins: a client gathers current context and keywords from open documents and loads related information from the server.

A view on the Semantic Desktop was given by Stefan Decker and Martin Frank in 2004, their paper called "The social semantic desktop" [8]. It focuses on the technology threads that are available and have to be combined to create the Semantic Desktop. The need for the system and the solutions it will provide are outlined. A possible roadmap is drawn, as mentioned above.

A major research project concerning an integrated approach in our field is the *Haystack* system by Quan et al. [15] 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. It is a replacement for many applications including word-processors, email clients, image manipulation, instant messaging and other functionality. They provide a complete semantic programming environment, from user interface to database. One disadvantage was that the prototype system had performance problems in 2003. These have been identified and addressed in the upcoming *Hayloft* project.

MyLifeBits by Microsoft Research is a lifetime store of multimedia data, based on the assumption that all information a single person reads and hears can soon be stored on a portable device. Every day a person consumes audio, video, text and other media. If a hypothetical disk of one terabyte per year is available, it would be possible to store all this multimedia on it. The MyLifeBits paper describes a concept how to manage this huge amount of media, how to classify and retrieve the data [16].

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 SmartTags and the Information Bridge Framework.

From the **open source scene**, several projects are aiming at a semantic desktop environment, one such a project is *Chandler* managed by the osa-foundation and lead by Mitch Kapor (who designed Lotus Agenda). It is a Personal Information Manager (PIM) intended for tasks like composing and reading email, managing an appointment calendar and keeping a contact list. It simplifies information sharing with others, and calls itself an *Interpersonal Information Manager*.

The *Fenfire* [17] project is at an earlier stage, dealing with the problem of visualising and 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 [18]. 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.

From these numerous examples, which only give starting points for the interested reader, we see that the field of the Semantic Desktop is already advanced and that different – sometimes competitive, other times complementary – approaches exist.

2.2 Tools

The active community produced a variety of tools that are used in the projects or that are end user applications. We will now categorize these tools based on a scheme similar to one developed for [19] and build a table that gives and overview, see table 1. Two main categories are assumed—first are the *grounding technologies*, the basic building blocks of system technologies and Semantic Web technologies. Second are *information interaction* tools providing users with interfaces to author and browse information. A third category are *ontologies* and ontology related tools.

The grounding technologies consist of storage, search and communication facilities. Storage and search are repositories that hold RDF and ontology data in a persistent way and to allow semantic search or fulltext search on the data. Known projects here are *Jena* [20], *Kowari* [21], *RDF Gateway* [22], or *Sesame* [23]. The support for full-text search is sometimes a feature of the repositories, if not it can be implemented through projects like *Apache Lucene*. For the Semantic Desktop we face several problems with repositories. First, multiple incompatible interfaces are implemented by the systems; therefore state that we need standardized interfaces for storage servers. The upcoming SPARQL standard [24, 25] will provide us with these. Although these repositories are in common use, they are far away from perfection. A description of problems with performance and ease of use can be found in the YARS project description by Harth et al. [26]; they tested the read and write performance of common open source RDF repositories and found major deficiencies. One store was not installable at all. *Communication technology* needed to receive and send messages is today provided by e-mail, instant messaging and peer-to-peer systems. It is possible to use these technologies to send semantic messages, as shown in [27] for semantic email or for the Jabber protocol (a standard for instant messaging) in the *Nabu* project [28], a semantically enhanced Jabber server. On the Semantic Desktop, these existing communication ways will be used to send semantic messages.

The shown storage, search and communication technologies will be used to store and communicate data that is expressed using *ontologies*. Users will work with several ontologies and the information expressed in these ontologies will come from heterogenous sources. A crucial factor will be the integration of ontologies by ontology mapping. Common ontologies we find on desktop computers today are Dublin Core, FOAF, iCalendar and more. We expect that through diversification and selection (an evolutionary, community process) a combination of many popular ontologies will be used on the future Semantic Desktop.

The user experience will be determined by *information interaction* software. Common applications here are ontology editors, domain specific applications, browsers and

personal knowledge management tools. *Protégé* by Stanford Medical Informatics is a popular ontology editor, *PhotoStuff* by the Mindswap group is a photo annotation tool. *Tidepool* is another photo editing tool, with a commercial background; together with the website *Storymill.com* users can annotate and publish photos. The many RSS readers that are available today can be seen as domain specific applications—they focus on news and information syndication. On the personal information management side, we find *Microsoft Outlook* or *Lotus Notes* in many companies. Another good example is FRODO Taskman [29] which realizes a fully RDF based semantic workflow engine. We expect semantic personal knowledge management tools in the future, that can integrate heterogenous sources taken from the Semantic Desktop.

3 How to build a Semantic Desktop

In this section we will describe how the parts for a Semantic Desktop can be assembled together and what new features have to be implemented. Starting with the new requirements that come with the Semantic Desktop and how these requirements can be fulfilled, we then move on to well known features that are already implemented. But before we go into details, we have to step back and take a look at the way people think and express their mental models, so that we understand how the Semantic Desktop can support this.

3.1 Respect personal mental models

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) [30].

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 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 thesis that all documents which are available on our individual workstation are somehow related to our **individual background**, to the ongoing tasks and running processes we are involved, or to our personal interests. Further, the documents capture information about concepts we make of the world: persons, places, projects, topics, etc. These concepts are highly subjective but can be expressed using basic application features like 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 like "brainfiler" [31, 32] or GATE [33]. Hence there exists an interaction between mental models and formal ontologies, mental models find their match in the formal, symbolic representation of ontologies.

Although the directories at individual workspaces are highly subjective, we take into consideration that collaborators usually have a **common background**. In [34] it is shown how a shared background and an awareness of a coworker'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 are formalized in RDF and preferably OWL and are used by the desktop application.

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 is the user doing, what was the user doing in the last hour, day, year; what are topics relevant to the peers and the company of the user; and much more can be used to capture this context.

We also see that the **context may switch**: while most of the work of a user 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. Then, the system becomes a supplement to the memory of the user by doing some of the knowledge management work. 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" [4]. The Semantic Desktop is application-independent. The software doesn't force the user to adapt and instead adapts to the user and not only that—it also adapts to other software employed by the user.

Respecting the personal mental models can be summarized as: **do not assume one application alone representing the ideas of the user, but manage the personal concepts of the user in cooperation with other applications.**

3.2 Context and user observation solutions

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 [35]

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.

The group around Wolfgang Nejdl published a paper on "Activity Based Metadata for Semantic Desktop Search" by Chirita et al. [36] 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 behavior 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 currently under evaluation at the DFKI in the eFisk project [37],[38] is to capture the reading behavior 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.

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.

3.3 Searching the Semantic Desktop

Barreau and Nardi [39] analyzed the searching behavior on desktop computers in 1995 and identified two different search strategies when users are looking for information that is stored on their desktop computers: first, a path search is done, looking into folders and directories that could contain the document. If the path search does not succeed, fulltext search strategies are used. Today, desktop search engines are a major market and tools like *Google Desktop*, *Apple Spotlight*, *Yahoo! Desktop Search* or *Microsoft Windows Desktop Search* are products in a competitive market. The features provided in these free tools are satisfying to most users but far behind what is state of the art in commercial tools like *Autonomy* or *Convera* do on a company level and what is proposed in current research papers.

We expect that sophisticated information retrieval techniques will find their way to the Semantic Desktop. In fig. 1 we show a few technologies that are expected to be available. Starting from today's fulltext search on the top-left we identify three directions towards Semantic Desktop search. In dark grey, to the top-right, text based technologies are listed. In light grey, to the lower-left, we see semantic technologies that benefit from metadata and explicit links between information items. These approaches are developed by Nejdl et al. [36, 40] and other researchers. Central are ontologies and

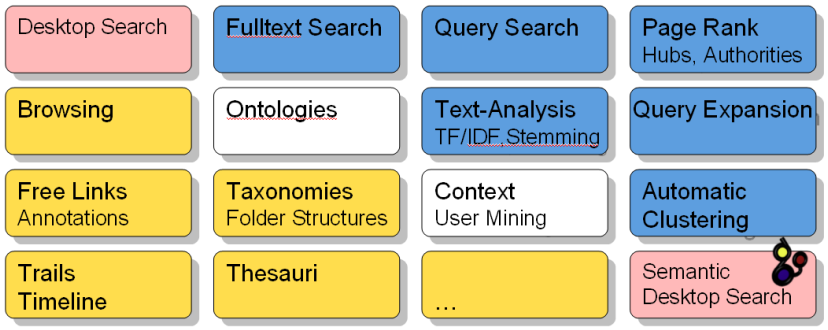


Fig. 1. Toward Semantic Desktop Search

context based approaches. In the figure we list some examples how to improve desktop search, but there are surely more ideas that contribute to the field. We intentionally left one field blank to represent the missing ideas. At the end, the combination of the listed technologies will improve the way users search, find, and experience information.

3.4 User interface

Looking at the building blocks mentioned above, we find similarities in the users interfaces and in the architecture that the software is build upon. We abstract now from the concrete examples and describe patterns we found in the user interfaces and architectures that are used today.

A typical interaction sequence in such an application is as follows:

- User searches and finds the information of interest using search services or by opening known resources via a path and confirms to edit/view it.
- Remote or local repository is contacted for the data. It usually drills down to one RDF graph and one current resource to view and edit.
- Additional data from ontologies is loaded to understand the data. Inference engines are used to augment the loaded information.
- Related information is gathered, using the loaded graph and the current resource as a starting point. Related information comes from remote and local repositories. Ontologies, thesauri, text similarity, and context are used to find related information.
- User browses information and makes decisions. New facts are entered and the personal mental model changes.
- User stores changed information to a local or remote repository

This program workflow itself is simple, and simplicity is a key feature of useful software. Systems that went beyond the simple workflow faced problems of complexity. For example, the *gnowsis* system started as a mixture of database, inference engine, user interface, and data integration architecture. The high goals of gnowsis lead to a complex architecture and performance problems which again forced us to refactor the project and split it into reusable components (a process that is not finished yet). *Haystack* also consists of database, user interface and domain specific (email, instant messaging, picture editing) functions. *Haystack* offers useful features and is a well administered project,

but the demands on computing power, memory and disk storage are high. Also, users faced with such complex systems need a long training time to understand the system and benefit from them.

Protégé gives an example of a clean architecture: provide a fast, extensible user interface for ontology editing and leave storage and inference to plugins and external services.

The following description gives a rough image what a typical Semantic Desktop application of today looks like. We expect totally new interaction models for the future that extend this model, as already the example applications extend the model in different ways. Visual examples are given in fig. 3. As a reference we took these applications: Mindraider, Gnowsis, Aduna Autofocus, Haystack, PhotoStuff, Protégé, Personal Brain (thebrain.com), Windows Vista.

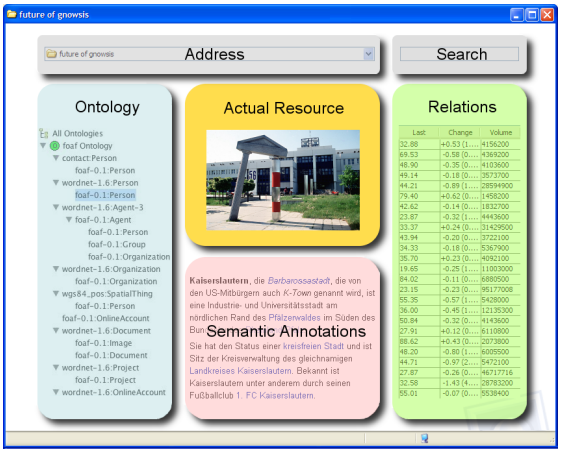


Fig. 2. a typical Semantic Desktop application user interface

We propose that the core parts of a user interface and application for information interaction are (see figure 2):

- An **adress bar** comparable to that of a web browser, where the user can easily enter the URI of the resource she wants to edit. Optionally, the address bar may also contain the address of a model/RDF graph that is currently edited.
- A single **fulltext search field** allowing searching for a resource like it is provided in Aduna Autofocus. Users expect that a plain text search field allow then to search on all possible resources and will, according to Nielsen [41], also demand such a search field.
- An visual area representing the **currently selected resource**. This is usually the center component and receives the focus of the user during editing. Visual feedback (color, font, etc.) about the currently selected resource is needed here.
- An area to add and change **annotations** of the currently selected resource. It may be part of the last point or a separate editor window. Such a component can be found,

e.g., in gnowsis or mindraider and will be provided in windows vista. Possible metaphors for it are wiki-like editors or forms.
- Additional **relations** of the current resource are also part of the user interface. These relations are often inferred based on factors like text similarity, related time or explicit links. Examples are given in mindraider, gnowsis, haystack or personal brain. They can help to ensure that all kind of information about a single resource can be presented to the user within a single window.
- An embedded **ontology browser** is also required. Respect that the ontologies are shared between applications and show the both the **personal concepts** and the **common ontologies** (as mentioned above). Users need the ability to relate the currently selected resource to the ontologies.

Because this kind of application would be monopolizing the user's attention, e.g. like an email programm, it would be best used in full screen mode. That leads to the conclusion that it has to be a sovereign posture program [42]. This is also enforced by the fact that such an application would be used very often and therefore dominates the users attention as a primary tool. The fact of having an sovereign posture points out that a semantic desktop application has to be designed for optimal use by perpetual intermediates (see axiom in [42, chapter 8]).

For future Semantic Applications, users will expect that the experience is similar to existing applications. An overview of existing applications is given in Figure 3. Based on the expectations of users we recommend: *when building Semantic Desktop applications, design the user experience in a way that can be recognized and understood by the users of today*.

3.5 Architecture of a Semantic Desktop application

Under the hood, we also find similarities in existing applications and generalize now to give the reader an insight to how today's applications are built. Separating user interface from database is a rule of thumb that can always help, the model–view–controller design pattern is also common. For Semantic Desktop applications, we find that a common pattern is to focus on the editing of a single resource, one after another, and support the usual actions of loading a file, editing it and storing it. In the semantic web, where the notion of files slowly shifts to the notion of RDF graphs, we propose an architecture that focuses on the editing of these graphs and resources inside graphs. The architecture, illustrated in 4, is aligned at the model-view-controller pattern:

- The *model* to show and manipulate is one RDF graph. It can be loaded from a local or remote repository and can also be stored remote or locally. Ontologies and related information are also models, but they are usually secondary data and seldom changed.
- The *controller* is application logic that is described using inference rules or program code. It is highly domain specific.
- The view (user interface) is already described above Fig. 2. It is also domain specific but conforms to common patterns.

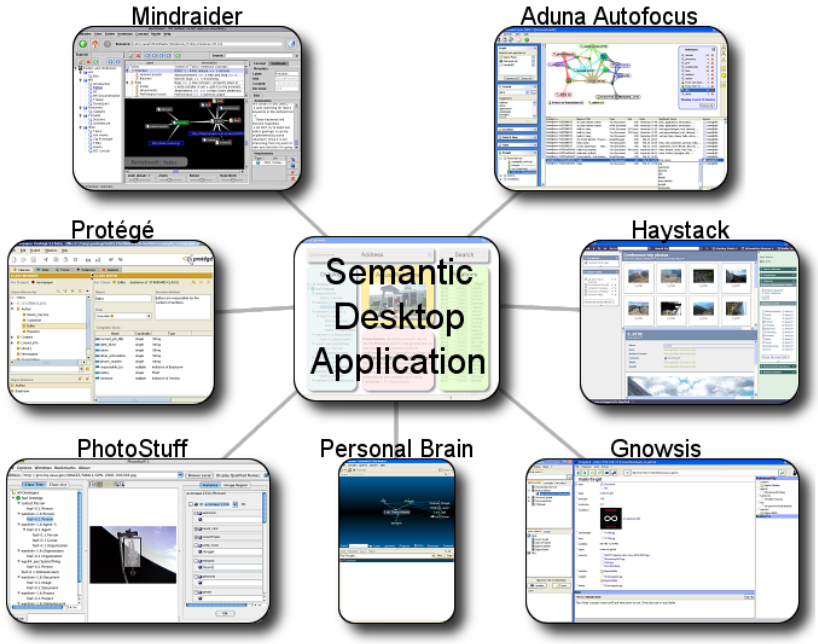


Fig. 3. Examples of existing user interfaces

3.6 Semantic Desktop enabling framework

To build information interaction applications which support above features, a basis architecture should be put in place. The diverse applications will need centralized services, so that not every application has to re-implement the wheel. These services will be part of a **framework** that runs as an invisible background server on a Semantic Desktop. Because they allow us to build user interfaces faster, we call them **enabling**.

Hence what services are elemental to a Semantic Desktop? This question is our concern in the *gnowsis.org* project, which serves as a prototype and test-bed for future applications. A few services are common technology today, the more complicated services are described below.

central RDF repository Even if the architectures differ – a central RDF repository is always there.

central search on the repository and documents a fulltext search and semantic (ontology aware) search service is needed

adapters It is agreed that existing data sources and applications have to be integrated. A detailed discussion on adapters can be found in [12].

ontologies The basis for information articulation and communication are common ontologies, their formal representation needs to be accessible to run inference and adapt user interfaces. We recommend to separate ontologies and make them available through dedicated services, so that developers clearly know what RDF graphs to use when the question comes to ontologies.

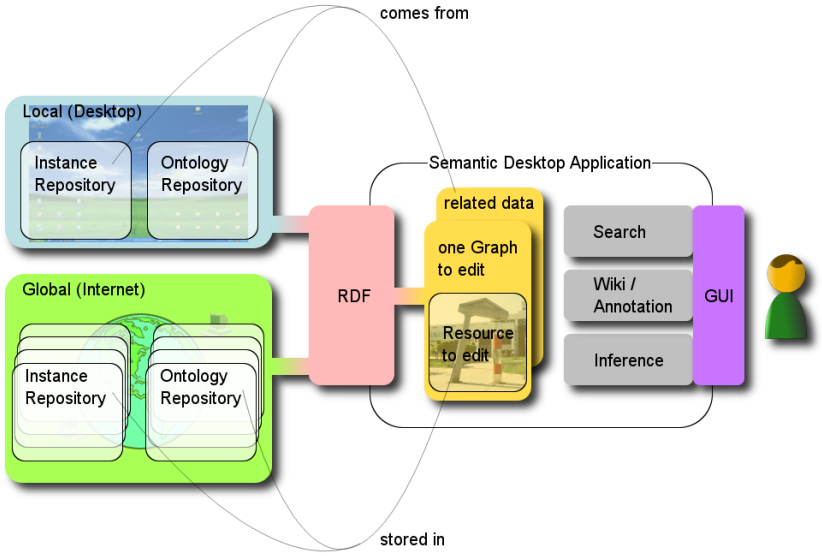


Fig. 4. A typical Semantic Desktop application architecture

context capture The quoted research projects suggest to observe the user behavior and user background via plugins to applications and the operating system. The plugins then report the actions relevant for the context to a service that stores the context and makes it available for other applications [35, 36].

pluggable architecture Service discovery and communication in the large are tasks tackled in the SWWS project[1]. Simpler structures can be used on the desktop suggested in [18].

These core capabilities are more or less available today. The next step will be to standardize their interfaces and provide stable implementations. The more difficult features (context and workflow) are still open but we expect to define the needed interfaces in the upcoming NEPOMUK project.

3.7 Merging the blocks—a Semantic Desktop

Above listing of existing and future developments leads to a description of an integrated system—the Semantic Desktop.

In Figure 5 an overview is given on how the building blocks of a Semantic Desktop work together. It is an evolution of the gnowsis architecture as described in [7]. The Semantic Desktop grows on a ground of data and information, the information is stored invisible to the user, in a database system or a RDF repository. The Semantic Desktop itself can now be seen as a tree – the roots of this tree are the stored information items and ontologies, invisible, under the surface, stored in semantic storage systems. Here we also find “grounding technology”. Above the soil are the applications visible to the user. They are independent from the tree but can use the tree to access the information in

[1] http://swws.semanticweb.org/

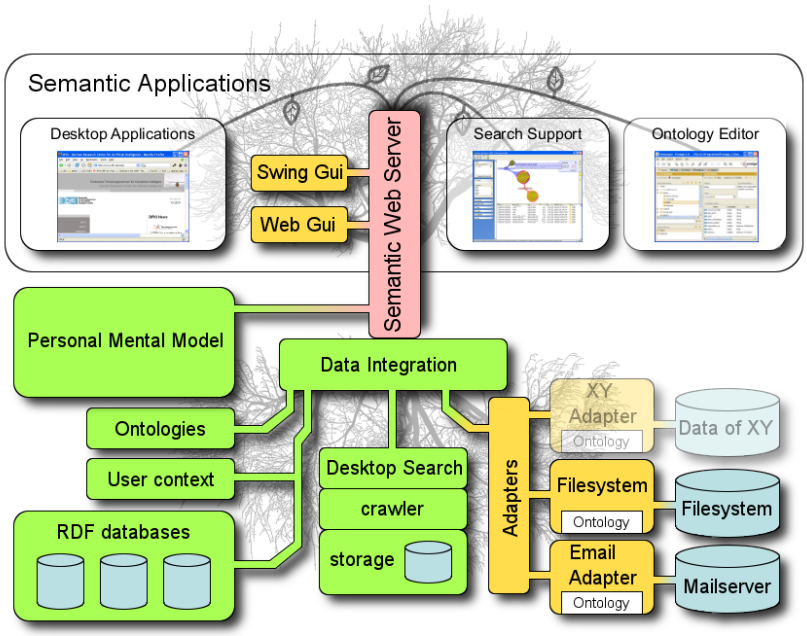


Fig. 5. Parts of a Semantic Desktop

the soil. The trunk of the tree, where it surfaces, consist of semantic web protocols and a server that gives access to the semantic services underneath. On this trunk, the branches and leaves grow, *information articulation and browsing* software. Applications can also connect to the tree and pick its fruits – use the information existing in the ground.

4 Summary and Outlook

The field we call today "Semantic Desktop" is both old (memex, hypertext systems and the web) and new (first publication with the term - 2003). A brief historical abstract was given, listing the projects and publications that form this field. The core idea is to bring Semantic Web technologies to the desktop, enabling people to use their desktop computers like a personal semantic web, where applications integrate and ideas are connected through ontologies. This idea was already addressed in several research projects and software products which are listed. We distinguished grounding technologies, ontologies and information interaction applications. These are the building blocks available today to build the Semantic Desktop of the future. We should now align our different ideas of the Semantic Desktop, for this we provided a definition of the term. We do not claim this definition to be final, but to be a starting point.

Also, we identified the need to standardize application programming interfaces and provide a background framework, that supplies enabling services. The user interface and the architecture of existing applications was presented and a view design patterns extracted, to provide developers with more indications where to start. The upcoming NEPOMUK project, proposed by a consortium of experts and lead by the DFKI, will

help building a community of experts that develop and use the Semantic Desktop. Part of the project are free open source implementations to standardize the interfaces, provide developers with example applications and end users with useful Semantic Desktop applications.

The Semantic Desktop will connect the semantic web to individual people, working on their desktop computers. It will allow them to write down ideas and knowledge and to share these ideas with others.

Acknowledgement We want to thank Dominik Heim from the FH Kaiserslautern for the graphic design of the included illustrations and for the comments on user interaction and visual metaphors.

References

1. Bush, V.: As we may think. The Atlantic Monthly **176(1)** (1945) p101–108
2. Freeman, E., Gelernter, D.: Lifestreams: A storage model for personal data. SIGMOD Record (ACM Special Interest Group on Management of Data) **25** (1996) pp80
3. Nelson, T.: As we will think. On-line 72 Conference Proceedings **vol. 1** (1972) pp. 439–454
4. Lee, T.B.: Weaving the Web, The Past, Present and Future of the World Wide Web by its Inventor. Texere, London (2000)
5. Berners-Lee, T.: Frequently asked questions by the press. (http://www.w3.org/People/Berners-Lee/FAQ.html)
6. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **89** (2001)
7. Sauermann, L.: The gnowsis-using semantic web technologies to build a semantic desktop. Diploma thesis, Technical University of Vienna (2003)
8. Decker, S., Frank, M.: The social semantic desktop. WWW2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web (2004)
9. Sauermann, L.: The semantic desktop - a basis for personal knowledge management. In Maurer, H., Calude, C., Salomaa, A., Tochtermann, K., eds.: Proceedings of the I-KNOW 05. 5th International Conference on Knowledge Management. (2005) 294 – 301
10. Dengel, A., Abecker, A., Bähr, J.T., Bernardi, A., Dannenmann, P., van Elst, L., Klink, S., Maus, H., Schwarz, S., Sintek, M.: Epos - evolving personal to organizational knowledge spaces (2002) Project Proposal, DFKI GmbH.
11. Harth, A.: Seco: mediation services for semantic web data. Intelligent Systems, IEEE **Volume 19** (2004) 66 – 71
12. Leo Sauermann, S.S.: Gnowsis adapter framework: Treating structured data sources as virtual rdf graphs. In: Proceedings of the ISWC2005. (2005)
13. C. Bizer, A.S.: D2rq-treating non-rdf databases as virtual rdf graphs. In: Proceedings of the 3rd International Semantic Web Conference (ISWC2004). (2004)
14. Corp., M.: Information bridge framework. (http://msdn.microsoft.com/office/understanding/ibframework/default.aspx)
15. Quan, D., Huynh, D., Karger, D.R.: Haystack: A platform for authoring end user semantic web applications. In: International Semantic Web Conference. (2003) 738–753
16. Gemmell, J., Bell, G., Lueder, R., Drucker, S., Wong, C.: Mylifebits: Fulfilling the memex vision. In: ACM Multimedia December 1-6, Juan-les-Pins, France. (2002) pp. 235–238
17. et al., B.F.: the fenfire project. (http://fenfire.org/)
18. Geldart, J.: Rdf without revolution an analysis and test of rdf and ontology. Bachelor thesis, Department of Computer Science, University of Durham (2005)

19. Lausen, H., Stollberg, M., Hernández, R.L., Ding, Y., Han, S.K., Fensel, D.: Semantic web portals - state of the art survey. Technical Report 2004-04-03, DERI (2004)
20. McBride, B.: Jena: Implementing the rdf model and syntax specification. In: Proc. of the Semantic Web Workshop WWW2001. (2001)
21. the kowari project. (http://kowari.sourceforge.net/)
22. Corp., I.: Rdf gateway semantic web server. (http://www.intellidimension.com/)
23. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: Proc. of the International Semantic Web Conference 2002. (2002)
24. (edt), K.G.C.: Sparql protocol for rdf. Working draft, W3C (2005) http://www.w3.org/TR/rdf-sparql-protocol/.
25. Prud'hommeaux, E., (edts), A.S.: Sparql query language for rdf. W3c working draft, W3C (2005) http://www.w3.org/TR/rdf-sparql-query/.
26. Harth, A., Decker, S.: Yet another rdf store: Perfect index structures for storing semantic web data with contexts. http://sw.deri.org/2004/06/yars/doc/summary last change Jan 2005, visit Aug 2005 (2005)
27. McDowell, L., Etzioni, O., Halevey, A., Levy, H.: Semantic email (2004)
28. Osterfeld, F., Kiesel, M.: nabu semantic jabber server. (http://nabu.opendfki.de)
29. Elst, L.v., Abecker, A., Bernardi, A., Lauer, A., Maus, H., Schwarz, S.: An agent-based framework for distributed organizational memories. In Bichler, M., Holtmann, C., Kirn, S., Müller, J.P., Weinhardt, C., eds.: Coordination and Agent Technology in Value Networks, Multikonferenz Wirtschaftsinformatik (MKWI-2004), 9.-11.3.2004, Essen, GITO-Verlag, Berlin (2004) 181–196
30. ACM: Acm classes 1998. (http://www.acm.org/class/)
31. AG, B.: the brainfiler text classification system. (http://www.brainbot.de)
32. Maus, H., Holz, H., Bernardi, A., Rostanin, O.: Leveraging passive paper piles to active objects in personal knowledge spaces. In: Proceedings of 3rd Conference Professional Knowledge Management: Experiences and Visions. (2005) 43–46
33. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: 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)
34. Clark, H.H.: Using language. Cambridge University Press (1996)
35. Schwarz, S.: A context model for personal knowledge management. In: Proceedings of the IJCAII'05 Workshop on Modeling and Retrieval of Context, Edinburgh (2005)
36. Chirita, P.A., Gavriloaie, R., Ghita, S., Nejdl, W., , Paiu, R.: Activity based metadata for semantic desktop search. (2004)
37. Miller, T., Agne, S., Dengel, A.: eFISK – eine aufmerksamkeitsbasierte Schlüsselwort-Extr aktions- und Information Retrieval-Maschine. Abschlussbericht 15202-386261/659, Stiftung Rheinland-Pfalz für Innovation (2005)
38. Miller, T., Agne, S.: Attention-based information retrieval using eye tracker data. In: Proceedings of the Third International Conference on Knowledge Capture (K-CAP05). (2005) To appear.
39. Barreau, D., Nardi, B.A.: Finding and reminding: File organization from the desktop. (1995)
40. Nejdl, W., Paiu, R.: I know i stored it somewhere - contextual information and ranking on our desktop. 8th International Workshop of the EU DELOS Network of Excellence on Future Digital Library Management Systems (2005)
41. Nielsen, J.: Mental models for search are getting firmer. http://www.useit.com/alertbox/20050509.html see also the tutorial on Fundamental Guidelines for Web Usability at the User Experience 2005 conference (2005)
42. Cooper, A., Reimann, R.: About Face 2.0 - The Essentials of Interaction Design. Whiley publishing inc (2003)

	Type	Today	Semantic Desktop
Grounding Technology	Storage	* Jena * Sesame * RDF Gateway	storage supports SPARQL and semantic protocols
	Search	* Lucene * Desktop Search Tools	semantic search services
	Communication	* Jabber, IM * email * P2P networks	semantic messaging and P2P
Ontologies		* DC * FOAF * iCalendar * SKOS * Thesauri * PIM	* popular ontologies * ontology mapping tools * desktop ontologies
Information Interaction	Ontology Editing	* Protege * IsaViz * KAON	ontology editors present in all applications
	Domain Specific	* Tidepool/Storymill * PhotoStuff * RSS Readers	Semantic Desktop Applications
	PIM and Workflow	* Microsoft Outlook * Lotus Notes * Frodo Taskman	Semantic PIM, Semantic Workflow

Table 1. Building Blocks

A Multi-Ontology Approach for Personal Information Management *

Huiyong Xiao Isabel F. Cruz

Department of Computer Science
University of Illinois at Chicago
{hxiao | ifc}@cs.uic.edu

Abstract. The increasingly huge volume of personal information stored in a desktop computer is characterized by disparate models, unstructured contents, and implicit knowledge. Aiming at a semantic rich environment, a number of *Semantic Desktop* frameworks have been proposed, concentrating on different aspects, including organization, manipulation, and visualization of the data. In this paper, we propose a layered and semantic ontology-based framework for personal information management, and we discuss its annotations, associations, and navigation. We also discuss query processing in two cases: query rewriting in a single personal information application, PIA, and that between two PIAs.

1 Introduction

In 1945, Vannevar Bush put forward the first vision of personal information management (PIM) system, Memex, by pointing out that the human mind "operates by associations", and we should "learn from it" in building Memex [4]. The Hypertext systems (see the survey of Conklin [6]), which flourished in the 80's, reinforced this vision and yielded the current World Wide Web, in a broader scope. Recently, with the Semantic Web vision [2], a number of PIM systems associated with that vision, hence called *Semantic Desktop*, have been proposed. By summarizing these proposals and taking into account the characteristics of personal information (PI), we propose the following principles that a PIM system should follow:

Semantic data organization. Almost all existing approaches are trying to go beyond the hierarchical directory model. The critical factors of semantic data organization include adequate annotations, explicit semantics, meaningful associations, and a uniform representation. A semantic-rich data organization has several advantages. First, the annotations and associations (as the *superimposed* information over the coarse data [19]) form the context of the PI, thus making the data more easily understandable. Second, the superimposed information also allows for finer and more flexible manipulation (e.g., browsing and querying) on the data. Third, an explicit formal semantics for the data can facilitate reasoning on the data and deriving new knowledge. Finally, the uniform representation can support the integration of data that may be heterogeneous.

Flexible data manipulation. A PIM system can provide integration, exchange, navigation, and query processing of the stored personal information. The framework of PIM,

* This work was partially supported by NSF Awards ITR IIS-0326284 and IIS-0513553.

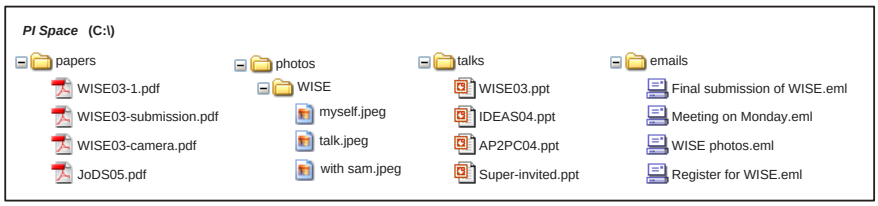


Fig. 1. An example of files in a PI space.

including the data model, query language, and user interface, should provide multiple ways to manipulate data in a powerful and flexible manner. Furthermore, a PIM system should possess the capability for seamless communication (or interoperability) with external sources (possibly in another PIM system), e.g., in a peer-to-peer (P2P) way [26].
Rich visualization. Multiple visualizations can help the user in understanding data. Instead of providing separate views of the data as most traditional applications do, a PIM system should support data visualization from different perspectives, to offer a comprehensive view. Examples include association-centric visualization [24] and time-centric visualization [13, 12].

Example 1. Figure 1 presents a fragment of PI space, which consists of four directories of files in the hard drive C:\. The papers directory contains four papers of the format pdf, photos\WISE contains three pictures taken at the WISE '03 conference, talks contains four Powerpoint files that respectively are slides of four talks, and emails contains four saved email messages. Even if the concrete contents of all these files are unknown, we can tell from their filenames that they are somehow related to each other. Unfortunately, their storage in different and possibly unrelated directories does not show their inter-relationships, thus resulting in possible difficulties in locating the wanted information. Some keyword-based searching techniques, e.g., offered by the Google Desktop Search,[1] can retrieve all files that are relevant to WISE. However, without further inspection of the content of each file, the user would not be able to discover certain associations between them, e.g., that file JoDS05.pdf is an extended journal paper of WISE03-camera.pdf.

From this example, we can see that the lack of semantic associations among the stored data could be a handicap for data and knowledge discovery. In this paper, we focus on issues of semantic data organization and management in PIM, by taking the following approach:

1) We propose a layered framework for PIM, in which multiple ontologies playing a variety of roles are employed. Specifically, the *resource layer* stores all the PI resources (using URIs), metadata of the PI, and all kinds of associations using RDF. The *domain layer* contains the ontologies specific to various domains that are used to structure the data and categorize the resources. The *application layer*, built on top of the domain layer, is where the user constructs different application ontologies for different purposes of data usage. The benefits of this layered architecture include: i) It provides a

[1] http://desktop.google.com

semantics-rich environment for personal information management; ii) The system provides flexibility and reusability, by decoupling the domain and application ontologies, so that the construction of application ontologies for different applications can reuse the underlying domain ontologies. We argue that this provides certain advantages over the use of a single domain model for all the PI (e.g., [9]).

2) We discuss in detail how to utilize superimposed information for semantic organization, focusing on the construction of resource-file and resource-resource associations. We also present the idea of *3D navigation*, which is a combination of the *vertical*, *horizontal* and *temporal* navigation in the PI space. The idea is inspired by some existing PIM systems including MyLifeBits [13] and Placeless Documents [10], and is demonstrated in a browser.

3) In our framework, the basic unit for the user to manage the Semantic Desktop is the *personal information application* (PIA). Each PIA aims to accomplish or assist a specific task (e.g., bibliography management, paper composition, and trip planning). The PIAs can be standalone, with their own application ontology, user interface, and workflows. Meanwhile, they can communicate with each other as if in a P2P network, by means of the connections (mappings) established between their application ontologies. In this sense, different PIAs interoperate at a semantic level. We describe query processing in our framework in two cases: within a single PIA or between two PIAs, in a P2P query processing mode.

Among the existing approaches to PIM, the Gnowsis project from DFKI[2] aims at a Semantic Desktop environment, which supports P2P (or distributed) data management based on *Desktop services* [26]. Like in our framework, Gnowsis uses ontologies for expressing semantic associations and RDF for data modeling. However, the emphasis of Gnowsis is more on the flexible integration of a large number of applications than on semantic data organization and manipulation. SEMEX [9] is another personal data integration framework that uses data annotation (i.e., schemas), similarly to our ontology-based framework. A single domain model is provided as the unified interface for data access.

MyLifeBits [13], Haystack [24], and Placeless Documents [10] are three PIM systems that support annotations and collections. Here, the concept of *collection* is essentially the same as the conceptualization (using ontologies) of resources in our framework. MyLifeBits supports easy annotation and multiple visualizations (e.g., detail, thumbnail, timeline, and cluster-time views on the data). For this purpose, the resources are enriched by a number of properties, including the standard ones (e.g., size and creation date) and more specific ones (e.g., time interval) [13]. Haystack aims to create, manipulate, and visualize arbitrary RDF data, in a comprehensive platform. For visualization, it uses an ontological/agent approach, where user interfaces and views are constructed by agents using predefined ontologies [24]. Placeless Documents introduces "active properties", where documents can have executable codes that provide document-based services [10].

Other approaches to PIM include Chandler,[3] Lifestreams [12], Stuff I've Seen (SIS) [11], and Xanadu [22], which focus on different aspects.

[2] http://www.dfki.de/web/

[3] http://www.osafoundation.org/

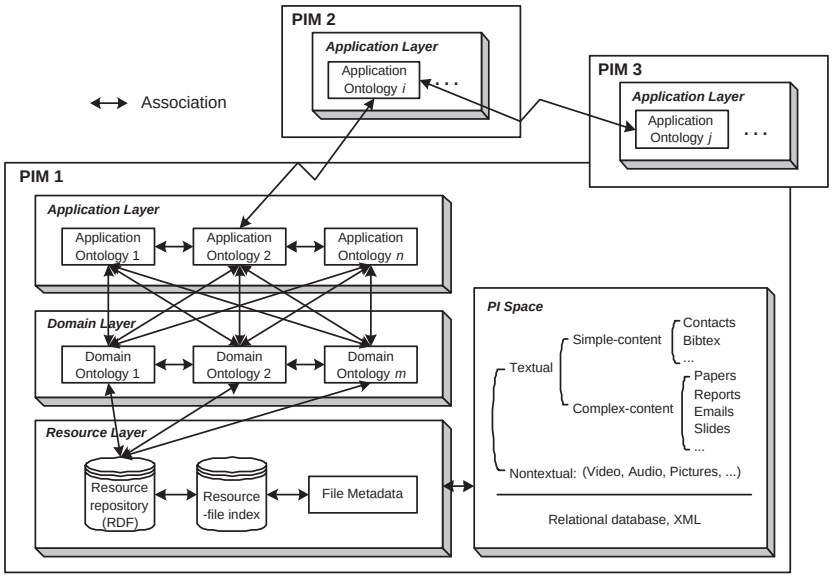


Fig. 2. An ontology-based framework of a PIM system.

The rest of the paper is structured as follows. In Section 2, we describe the layered framework and its main components. The semantic organization of the PI, including its approaches to annotation, association, and representation, is discussed in Section 3. Section 4 and Section 5 focus on two main ways of data manipulation, namely, navigation and query processing. Finally, we conclude in Section 6.

2 Framework

Our framework follows the principle of superimposed information, i.e., data or metadata "placed over" existing information sources [19]. This concept seems particularly useful for the organization, access, interconnection, and reuse of the information elements. We propose for PIM a layered ontology-based framework, as shown in Figure 2, with the following data components:

Personal information space. The personal information space may contain structured data (e.g., relational), semi-structured data (e.g., XML), or unstructured data. Unstructured data can be textual or non-textual (e.g., video, audio, or pictorial). Furthermore, textual files can be classified as simple-content or complex-content. More specifically, simple-content files have no references to other files. Typical examples include people contacts and Bibtex entries. In contrast, complex-content files have a flexible scheme of presentation, and may contain references to other files, e.g., by means of citations or hypertext links [6]. For example, a paper in the PI space may cite another paper (existing in the PI space or an external space), which, in turn, could cite other papers.

File description. We annotate each file using a file description (or metadata) consisting of a set of properties of the file. Each item in the file description is a property-value

pair. The file description is the first-level (direct) annotation for the individual files, and has the same scheme (structure) for the same type of files. For example, the following fragment contains a typical description of a JPEG file.

```
Dimensions:   3072 × 2048 pixels
Device make:  Canon
Color space:  RGB
Focal Length: 75
......
```

Domain ontologies. A number of ontologies are published on the Web. Examples of such ontology libraries include DAML Ontology Library,[4] the Semantic Web Ontologies,[5] and the Protégé OWL ontologies.[6] The ontologies in these libraries are typically designed and organized for different domains such as Conference, Person, Photo, and Email. In our framework, the domain ontology layer is designed to be loosely-coupled with the other layers, to enable the insertion and removal of ontologies as "plug-ins".

Resource-file index and RDF repository. One of the roles of domain ontologies is to provide the basis for data classification. In order to establish the connections between the files and the concepts in the domain ontologies, we treat each file as a resource, which is then classified as an instance of one or more concepts. The resource-file index is a local database storing these connections between resources and files. Furthermore, the various types of associations among resources (as instances of *association of concepts* in the domain ontologies) are stored in an RDF repository. The resource-file index and the RDF repository are both in the *resource layer*, providing resource instances for the domain ontologies in the *domain layer* above.

Application ontology. Above the *domain layer* is the *application layer*, which contains the ontologies for different applications. The domain ontologies, as an intermediate layer between the applications and the data, are meant to enhance the reusability and flexibility of the framework. More specifically, the application ontologies are defined as views of the domain ontologies, which can be reused for the construction of different application ontologies. In our framework, each *personal information application* (PIA), is associated with an application ontology, has access to relevant data, and is functionally independent of other applications. It may be infeasible to have a single ontology to cover various applications, e.g., for trip planning and paper writing. Instead, as many PIAs as needed can be designed in one or more PIM systems, where the PIAs can interoperate (e.g., through P2P query processing) for the purpose of integrating relevant information. This issue is elaborated on in Section 5.

Besides the data components described above, a PIM system also needs some functional components to perform all kinds of data and metadata processing, to make the framework work as a whole. Such components include an *indexer* (for establishing and managing the indexes of the files), a *wrapper* (for identifying and extracting resources from the files), and an *ontology designer* (for importing and editing an ontology). Because of space limitations, we do not elaborate further on these components.

[4] http://www.daml.org/ontologies/

[5] http://www.schemaweb.info

[6] http://protege.stanford.edu/plugins/owl/owl-library/

3 Semantic Data Organization

The layered architecture of our PIM framework described previously enables the reusability and the organization of semantically rich data for PIM. In this section, we discuss in detail the mechanisms that our framework uses to support the semantic organization of the PI space, including those for semantic annotation, association, and representation.

3.1 Annotation

Given that the data in the PI space is the base information, all the other data components in our framework are actually superimposed information over this base. The most fundamental function of the superimposed information is to provide semantic annotations of the base information to enable powerful and accurate data access. We discuss the following two aspects:

File description. It is especially important to provide the searcher with a detailed description of the nontextual files. When performing a keyword-based searching, the searcher matches the submitted keywords (e.g., "Canon") or key-value pairs (e.g., "Maker:Canon") with the property-value pairs of the file description, to find the right files requested by the user. Even for textual files, taking into account such metadata will improve the effects of full-text searching.

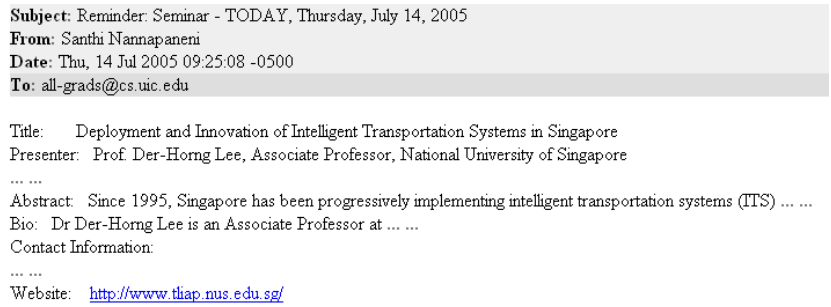
Domain ontologies. Given that a file is identified as a resource, we are able to annotate the file using a domain ontology, by associating the resource with a concept of an ontology. The domain ontology provides not only a context for understanding the data, but also semantic clues for the precise data retrieval. For example, the user can query the PI using a query language for RDF instead of using keywords. We note that a file can be an instance of more than one concept, according to different classification criteria.

3.2 Association

In our framework, semantic associations are used to relate all the data (base information) and metadata (superimposed information). There are two classes of associations: the *resource-file associations* that are actually the resource-file indexes and the *resource-resource associations* that are instances of the domain ontologies and are stored in the RDF repository.

Resource-file associations. In addition to the ontological resources that are used to identify (through data classification) the files, a (textual) file may contain and refer to a number of resources. Therefore, the resource-file associations can be classified into three kinds, such as *identification*, *containment*, and *reference*.

Example 2. Suppose that the user has saved an email message, which is an announcement of a seminar, as shown in Figure 3. First, the email can be classified as an instance of the concept Email, provided that the concept exists in some domain ontology. Then, the system can generate for the concept SeminarAnnouncement and its properties a new instance (i.e., resource), which is associated with the saved email by the relationship *containment*. Finally, a *reference* association can be established between the resource http://www.tliap.nus.edu.sg/ (e.g., of the concept WebsiteAddress) and the email message.



Subject: Reminder: Seminar - TODAY, Thursday, July 14, 2005
From: Santhi Nannapaneni
Date: Thu, 14 Jul 2005 09:25:08 -0500
To: all-grads@cs.uic.edu

Title: Deployment and Innovation of Intelligent Transportation Systems in Singapore
Presenter: Prof. Der-Horng Lee, Associate Professor, National University of Singapore
... ...
Abstract: Since 1995, Singapore has been progressively implementing intelligent transportation systems (ITS) ... ...
Bio: Dr Der-Horng Lee is an Associate Professor at ... ...
Contact Information:
... ...
Website: http://www.tliap.nus.edu.sg/

Fig. 3. An example of an email message.

The process of setting up the resource-file associations is the one of recognizing resources from the file description and/or the file content and then mapping them to the ontological concepts. The user may be concerned with the degree to which the resources should be extracted from a file and its description. For instance, in the previous example, the user can further create resources for the title and abstract of the seminar, and for the biography of the presenter. It is expected that this process (as well as the process of discovering resource-resource associations, as discussed later) can be maximally automated, to reduce the user's burden. For this purpose, we may utilize the following methods:

- **Keyword extraction**. From the text of a file, keywords can be extracted based on a thesaurus or be highlighted manually by the user. Each keyword can be taken as a resource contained by the file. The matching of the resources with the concepts in the domain ontologies can be guided by a thesaurus such as WordNet.[7]
- **Hyperlink analysis**. For the textual files that include hyperlinks to classified resources (e.g., a citation of a paper or a link to a webpage), we create for each hyperlink a reference-type resource-file association, as well as a resource-resource association between the referring resource and the referred one.
- **Natural language processing**. We can utilize the techniques (e.g., [1]) to parse each sentence of a text or its summary obtained by means of text summarization [20]. For each resulting triple $\langle subject, predicate, object \rangle$, we try to match it with the patterns $\langle s, p, o \rangle$ in the domain ontologies, where p is a property of the concept s and has a value typed of o. If such pattern exists, a resource-resource association of the form $\langle subject, predicate, object \rangle$ is then generated.
- **History**. As the framework proceeds with such classification and cognition, more and more knowledge about this process can be accumulated and reused by a new process.

Resource-resource associations. We borrow from the Object Oriented Design (OOD) techniques the following four types of relationships between objects: *instantiation* (i.e., membership), *association*, *aggregation* (i.e., whole/part), and *generalization* (i.e., inheritance). These four relationships, which are used in object models, are adopted to

[7] http://wordnet.princeton.edu

Table 1. Resource-resource associations.

Resource-resource associations	Intra-domain	Inter-domain	Intra-application	Inter-application	Domain-application
aggregation	√	√	√		
association	√	√	√		
instantiation	√	√	√		
generalization	√	√	√		
ontology mapping		√		√	√

describe the associations among concepts as well as resources in our framework. Note that the "association" in the above four types of OO relationships is distinct from the "association" in what we call resource-resource associations. In ontology terms, the former is actually projected to be the user-defined properties, e.g., writes can be a property of the concept Author, connecting to the concept Book. Table 1 summarizes the resource-resource associations in our framework.

By using the previously described techniques, we can discover the resources and their associations implied in the PI, and classify them into the domain ontologies, thus populating the ontologies. In the example of Figure 3, it is possible to extract a pattern ⟨Singapore, implements, ITS⟩, which could then be classifier as an instance of an ontological pattern such as ⟨Organization, implements, System⟩, where Organization and System are two concepts, and implements is a property. Note that the user is allowed to choose the granularity of this knowledge (resource and associations) discovery process, ranging from only taking the whole file as a single resource to analyzing the detailed contents of the file.

In addition, ontology mappings may be established between correspondences that connect concepts in different domain and application ontologies. Currently, we consider *equivalence* as the only semantics for the mapping between two concepts, although richer semantics of the mappings could be considered [17].

3.3 Representation

In our framework, all information, including file descriptions, the resources in the repository, and the resource-file indexes, are represented in the Resource Description Framework (RDF),[8] a W3C proposed standard. For the schema of these data (i.e., the application and domain ontologies), we use the vocabulary language for RDF, RDF Schema (RDFS).[9] The RDF model is a semantic network, where the nodes denote the resources and the edge are properties that represent the relations between resources. The network can also be seen as a set of statements (triples) in the form of (subject, predicate, object). RDFS is used to define the vocabulary (in terms of classes and properties) of the RDF data, such as rdfs:Class, rdf:Property, and rdf:type. Table 2 summarizes the RDFS vocabularies that are used to represent different types of associations.

[8] http://www.w3.org/RDF/
[9] http://www.w3.org/TR/rdf-schema

Table 2. RDF properties for the associations.

Relationship	RDF property	Comments
aggregation	rdfx:contains	rdfx is the abbreviation of the namespace, where the property contains is defined. For example, <#a, rdfx:contains, #b> means that a contains b.
association	User-defined properties	For example, <#wise03talk, presentedBy, #xiao> means that wise03talk is connected to xiao by the association presentedBy.
instantiation	rdf:type	For example, <#xiao, rdf:type, #Person> means that the resource xiao is an instance of the concept Person.
generalization	rdfs:subClassOf	rdfs:subPropertyOf is used for property generalization.

The use of RDF as the data model and RDFS as the ontology language in our framework is motivated by the nature of the RDF/S as a Web resources description mechanism and the fact that the PI is represented as a set of interrelated resources. In contrast, XML is not chosen because it cannot represent semantic associations [8]. Certainly, OWL (Web Ontology Language), as built on top of RDFS, is more expressive for ontology representation. However, the use of a slightly extended version of RDFS is adequate for our case of the resource-file and resource-resource associations.

The extension to RDFS is the following: we define in a namespace (abbreviated using the prefix rdfx) a new RDF property, contains, which is used to represent the *aggregation* relationship. For the representation of the *instantiation* and *generalization* relationships, we use rdf:type and rdfs:subClassOf, respectively. The *association* relationship is represented naturally by the RDF properties defined in the user-defined namespace. Figure 4 gives a concrete example of an RDF/S representation.

4 Semantic Navigation

It is critical for a Semantic Desktop to provide the user with the capability to access the stored data in a variety of ways. The user may want to browse the information by means of the flexible and intelligent navigation in the information space, including the base and superimposed information. The user may also desire that certain query facilities (e.g., keyword-based searching or certain query languages) be provided by the framework. In this section, we discuss the navigation in the data space of a Semantic Desktop. Query processing is discussed in the next section.

The semantic data organization in our framework enables the navigation in the PI space, making use of useful hints (e.g., the context of a concept being browsed) so as to facilitate the user's understanding of data. More specifically, by taking into account the layered architecture, the semantic navigation in our framework can be performed in three directions: (1) In *vertical navigation*, the user follows a path across layers. Two cases are possible for this way of navigation: top-down from the application ontologies to the stored files and bottom-up from the stored files to the application ontologies. (2) In *horizontal navigation*, the user follows links of concepts (or resources) within one layer. Typically, there are three cases of horizontal navigation, corresponding to each

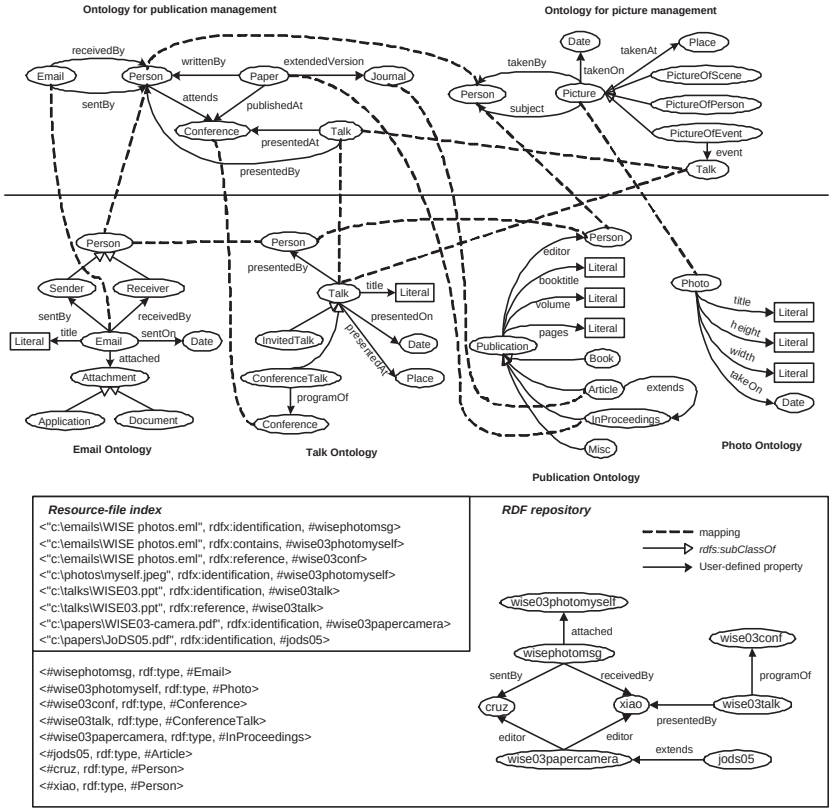


Fig. 4. Representation of the application, domain, and resource layers. All ontologies are represented in RDF/S. Two application ontologies for PIAs, i.e., picture management and publication management, are constructed. Below them are four ontologies for the domains of Email, Talk, Publication, and Photo, respectively. At the bottom, the resource-file and resource-resource associations are represented as triples or in a graph.

layer: application-to-application navigation, domain-to-domain navigation, and file-to-file navigation. (3) In *temporal navigation*, the user can navigate by following references in chronological order, each being a resource for the same real world object with a time stamp associated with it. For example, the user may want to look at different versions of a research paper.

All the base and superimposed information in the framework forms a directed graph, where the vertices are the resources in the ontologies and the files stored in the PI space, and the edges are the associations between the resources and files. We say that the three directions of navigation together provide the capacity of a *3-dimension* (3D) navigation mechanism, which can facilitate the construction of a browser. For instance, suppose the user is browsing a specific application ontology in a visualized browser. When the user clicks on the node of a concept in the ontology, the browser can then choose to

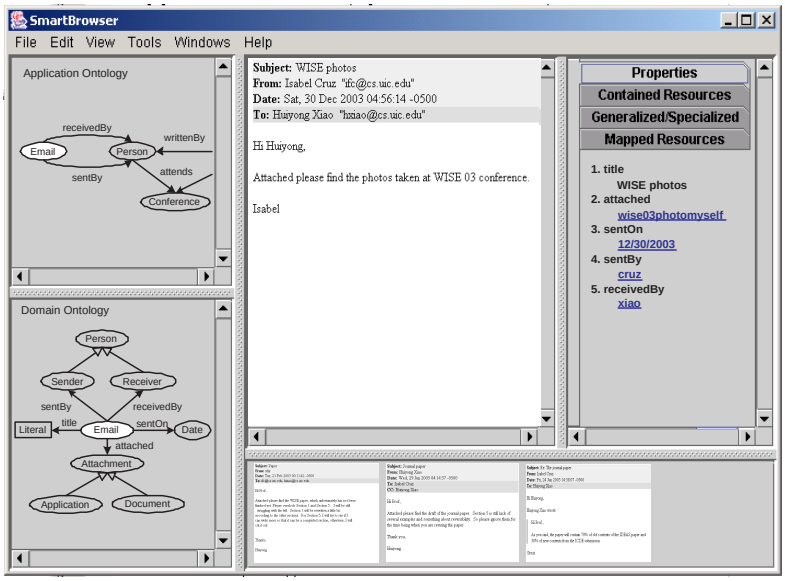


Fig. 5. The browser for PIM.

display the instances of the concept thus selected (by vertical navigation), the context of the concept in the domain (also by vertical navigation), and the associated concepts in other application ontologies (by horizontal navigation). Compared to the traditional navigation approach that is based on hierarchical directories, 3D navigation is based on semantic associations, similarly to those that humans establish between concepts.

Example 3. Consider the scenario shown in Figure 4. The spirit of 3D navigation is demonstrated in the browser of Figure 5. Suppose the *current* resource (file) that the user is browsing is an email message (i.e., wisephotomsg), which has some photos attached, which were taken at WISE '03. The concepts that this resource belong to are highlighted (in white) so as to show the contexts to which they belong. All associated resources are categorized and shown on the right tabbed pane, which provides a guidance for the user in navigating the PI space. The bottom-right pane shows the timeline of different versions of the current resource (if they exist) or all the resources belonging to the same concept as the current resource.

5 Semantic Query Processing

Unlike navigation, which is an interactive process, query processing is performed without further intervention from the user. To retrieve relevant data from the PI space, the user's request may be posed as a sequence of keywords or as a query formulated in a certain query language.

The keyword-based search matches the input keywords and the vector of words in the *candidate* documents, calculates the *similarity* for each of the matches, and returns

to the user the results after *ranking* them [25]. The results of a search are usually evaluated using the statistical criteria such as *precision*, *recall*, or a combination of them. The shortcoming of keyword-based search is that the semantic associations between relevant data are not considered. In contrast, query languages can provide a semantically richer access interface, thus facilitating the data retrieval and improving the accuracy of the answers. However, a query is usually performed based on an exact match between the query and the data, so that the recall of the answers is influenced, in the sense that some relevant but not matched data is not retrieved.

Since the two approaches complement each other, it is desirable to provide both of them. In this section, however, we mainly focus on query processing in our framework. We choose to express the queries in RDQL [16]; they can query both the resources and their associations. We discuss how to process a query submitted by the user in two cases: within a PIA and across different PIAs.

5.1 Query processing in a PIA

In our framework, the user query is formulated in RDQL (RDF Data Query Language), which uses an SQL-like syntax [16]. To reduce the user's burden, a graphic means can be used to facilitate the user's query formulation. For simplicity, we use a subset of RDQL, called *conjunctive RDQL* (c-RDQL), which can be expressed as a conjunctive formula: $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n})$, where $\boldsymbol{X_i} = (x_i, x_i')$ and p_i is an RDF property of x_i having the value x_i'.

In our framework, an application ontology is constructed over one or more domain ontologies, and the files in the PI space are formalized as instances of the concepts in the domain ontologies. If we consider the application ontology as the global ontology (since the user query is posed on it), the whole system can be seen as a GaV data integration system [18]. Therefore query processing in a single PIA is performed as in a GaV system. In particular, when the user poses a query (in RDQL) over the application ontology, the RDQL query is then rewritten into a new RDQL query in terms of the domain ontologies, based on the mappings between the global ontology and domain ontologies. By executing the rewritten query on the corresponding domain ontologies, resources (files) that match the query are then returned as answers to the query.

There are a number of algorithms for query rewriting in relational or XML data integration systems [14]. In a GaV based integration system, query processing is performed using a "unfolding" strategy [18]. More specifically, for rewriting a query (e.g., a conjunctive query) that is posed on the global schema or ontology, we simply substitute the predicates in the body of the query with the corresponding view definitions. In our framework, where the mappings between the application ontology and the domain ontologies are expressed as RDF class or property correspondences, the algorithm for query rewriting is similar to this strategy.

By assuming that there are no integrity constraints over the application ontologies and the user queries are formulated in c-RDQL, we give the formal description of our query rewriting algorithm in a single PIA, which we call ADREWRITING (for rewriting from Application ontologies to Domain ontologies), as follows. We note that we do not consider the namespaces of ontologies for simplicity of the description.

Algorithm ADREWRITING
Input:1. q_1 over the application ontology $\mathcal{G}$: $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_m(\boldsymbol{X_m})$;
2. $\mathcal{M}$: the mapping table between $\mathcal{G}$ and domain ontologies $\mathcal{S}_1, ..., \mathcal{S}_n$.
Output: q_2: A c-RDQL query over $\mathcal{S}_1, ..., \mathcal{S}_n$.
1. $head_{q_2} = ans(\boldsymbol{X})$; $body_{q_2} = null$;
2. **For** $i = 1$ **to** m **do**
3. (c_1, c_2) = name of the classes referred to by (x_1, x_2), for $\boldsymbol{X_j} = (x_1, x_2)$;
4. Search $\mathcal{M}$ to find (d_1, d_2) such that $\{(c_1, d_1), (c_2, d_2)\}$ are two class correspondences in $\mathcal{M}$;
5. Traverse $\mathcal{S}_1, ...,$ and $\mathcal{S}_n$ by following all kinds of associations, to find the vertices, $v_1, ..., v_k$, connecting from d_1 to d_2;
6. **If** $k = 0$ **then** add $p(x_1, x_2)$ (or $p(x_2, x_1)$) to $body_{q_2}$, if there exists p connecting d_1 to d_2 (or d_2 to d_1);
7. **Else for** $j = 1$ **to** $k - 1$ **do**
8. Add $p(\hat{x}_j, \hat{x}_{j+1})$ (or $p(\hat{x}_j + 1, \hat{x}_j)$) to $body_{q_2}$, if p is not a mapping and connects v_j to v_{j+1} (or v_{j+1} to v_j);
9. Add $p(x_1, \hat{x}_1)$ (or $p(\hat{x}_1, x_1)$) to $body_{q_2}$, if p is not a mapping and connects d_1 to v_1 (or v_1 to d_1);
10. Add $p(\hat{x}_k, x_2)$ (or $p(x_2, \hat{x}_k)$) to $body_{q_2}$, if p is not a mapping and connects v_k to d_2 (or d_2 to v_k);
11. $q_2 = head_{q_2}$:- $body_{q_2}$;

Example 4. Suppose the user wants to list all conference papers with their authors and journal version, using the query q_1 : $ans(x, y, z)$:- $writtenBy(x, y), extendedVersion(x, z)$, which is posed on the application ontology of publication management. For the variables (x, y, z), we get the classes that they refer to as (Paper, Person, Journal), as indicated by Line 3. By looking into $\mathcal{M}$, we find the corresponding class sequence as (Publication:InProceedings, Publication:Person, Publication:Article), where the names before the colons are domain ontology names. From Lines 5 to 10, we compute the predicates in the body of q_2 as follows.

q_2: $ans(x, y, z)$:- $editor(x, y), extends(z, x)$

By executing q_2 over the RDF repository as shown in Figure 4, we get the answer {(#wise03papercamera, #xiao, #jods05), (#wise03papercamera, #cruz, #jods05)}.

5.2 A2A query processing

Application to application (A2A) query processing occurs when an application is attempting to retrieve relevant data from another semantically related application, to answer a query. If the PIAs are considered as connected peers (i.e., service providers for certain data access), the A2A query processing is similar to that in peer-to-peer (P2P) systems [7, 15]. Whether the PIAs exist in a single desktop or are physically distributed makes no differences to the A2A query processing.

A2A query processing consists of two steps of query rewriting. First, we rewrite the original query q, which is posed on the application ontology $\mathcal{G}_1$, to a query q' on the other application ontology $\mathcal{G}_2$, according to the mappings between $\mathcal{G}_1$ and $\mathcal{G}_2$. Then, q' is rewritten to a query q'' on the domain ontologies, to which $\mathcal{G}_2$ is mapped. Answers are obtained by executing q'' on the RDF repository. The second query rewriting is exactly the one described by the algorithm ADREWRITING, whereas the first rewriting is slightly different from ADREWRITING. In particular, unlike the total mapping from an application ontology to the domain ontologies, some of the concepts in $\mathcal{G}_1$ may not be mapped to those in $\mathcal{G}_2$. Therefore, the answers returned by q'' may contain null values or Skolem functions for the unmapped concepts or properties.

The A2A mappings can be derived by composing the mappings between $\mathcal{G}_1$ and the domain ontologies, inter-domain mappings, and those between $\mathcal{G}_2$ and the domain ontologies. To evaluate both query rewriting processes, we need to check the equivalence (or containment) between a query and its rewriting. A *correct* query rewriting is the one that is equivalent to (or maximally contained in) the query. These two issues (*reasoning on mappings* [3, 23] and *reasoning on queries* [5, 21]) have been extensively studied and are beyond the scope of this paper.

6 Conclusions and Future Work

In this paper, we present our design of a PIM system. We propose a layered ontology-based framework, which aims to provide a semantics-rich environment for personal information organization and manipulation. The multiple ontologies existing in different layers of the architecture explicitly support the data semantics. Furthermore, the decoupling of the domain layer and the application layer enhances the flexibility and reusability of the framework. Specifically, we discuss in detail the semantic-enriched data organization, including the use of file descriptions and domain ontologies as annotations, and the construction of resource-file and resource-resource associations. We also introduce the idea of *3D navigation*, which is used in a desktop browser. We discuss query processing in our framework in two cases: within a single personal information application, PIA, and between two PIAs, using application to application (A2A) communication. A formal query rewriting algorithm is presented for the single PIA case.

In the future, we will continue the study and implementation of our framework. It is clear that a lot of the success of PIM systems lies on the successful automation of the different mechanisms that are needed. In particular, we will look further into the automation of the conceptualization of full-text files and that of matching resources to ontological concepts. Also, we will elaborate on the idea of 3D navigation both by studying a model for temporal navigation and by carrying out user studies. The study of A2A communication, including data exchange, collaboration, and query processing will also be continued. While RDQL queries are expressive, they may not be suitable for most users. We are therefore exploring visual queries that can express a class of RDQL queries "appropriate" for the semantic desktop.

References

1. M. Berland and E. Charniak. Finding Parts in Very Large Corpora. In *ACL*, 1999.

2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
3. P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.
4. V. Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, 1945.
5. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based Query Containment. In *PODS*, pages 56–67, 2003.
6. J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–41, 1987.
7. I. F. Cruz, H. Xiao, and F. Hsu. Peer-to-Peer Semantic Integration of XML and RDF Data Sources. In *AP2PC*, July 2004.
8. S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.
9. X. Dong and A. Y. Halevy. A Platform for Personal Information Management and Integration. In *CIDR*, pages 119–130, 2005.
10. P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton. Extending Document Management Systems with User-specific Active Properties. *ACM Transaction of Information System*, 18(2):140–170, 2000.
11. S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've Seen: A System for Personal Information Retrieval and Re-use. In *SIGIR*, pages 72–79, 2003.
12. E. Freeman and D. Gelernter. Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, 25(1):80–86, 1996.
13. J. Gemmell, G. Bell, R. Lueder, S. M. Drucker, and C. Wong. MyLifeBits: Fulfilling the Memex Vision. In *ACM Multimedia*, pages 235–238, 2002.
14. A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB J.*, 10(4):270–294, 2001.
15. A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *WWW*, pages 556–567, 2003.
16. HP Labs. RDQL - RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm, 2005.
17. Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: the State of the Art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
18. M. Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, Madison, Wisconsin, June 2002. ACM.
19. D. Maier and L. M. L. Delcambre. Superimposed Information for the Internet. In *WebDB*, pages 1–9, 1999.
20. I. Mani. Recent Developments in Text Summarization. In *CIKM*, pages 529–531, 2001.
21. T. D. Millstein, A. Y. Halevy, and M. Friedman. Query Containment for Data Integration Systems. *Journal of Computer and System Sciences*, 66(1):20–39, 2003.
22. T. H. Nelson. Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-use. *ACM Computer Surveys*, 31(4es):33, 1999.
23. N. F. Noy. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record*, 33(4):65–70, 2004.
24. D. Quan, D. Huynh, and D. R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In *ISWC*, pages 738–753, 2003.
25. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
26. L. Sauermann. The Gnowsis Semantic Desktop for Information Integration. In *The 3rd Conference on Professional Knowledge Management*, pages 39–42, 2005.

DocuWorld — A 3-D User Interface to the Semantic Desktop

Katja Einsfeld[1], Achim Ebert[1], Stefan Agne[2], and Bertin Klein[2]

[1] Visualization Lab, Technical University Kaiserslautern
P.O. Box 3049, 67653 Kaiserslautern, Germany
{k_einsfe,ebert}@informatik.uni-kl.de
[2] German Research Center for Artificial Intelligence (DFKI GmbH)
P.O. Box 2080, 67608 Kaiserslautern, Germany
{stefan.agne,bertin.klein}@dfki.de

Abstract. With the aim of replacing the 20-year-old 2-D desktop metaphor with a more natural, intuitive to use, and immersive 3-D interface, we developed a prototype application that allows the dynamic exploration of document collections. Document metadata and relations between documents are visualized with the help of information visualization techniques. The general visualization and navigation metaphor called *Thought Wizard Metaphor* allows user- and context-sensitive adaption of visualization modes and visualization environments. The automatic document organization modes provide different insights in document semantics and search results. These possibilities are supplemented by the possibility to store documents at user-defined, hierarchically organized locations, which allows quick relocation using spacial memory.

1 Introduction

Conventional document or file system viewers like the Windows Explorer or KDE's Konqueror are quite unintuitive to computer newcomers who know nothing about file system architecture. They sometimes have problems understanding where they "are" in the file system and remembering where their documents are stored. But advanced computer users with huge document collections stored in different hierarchical folders also suffer from the disadvantages of conventional file system viewers: they have to use their cognitive energy remembering names and locations of documents, navigating though the file system hierarchy, opening and closing folders, and looking through alphabetically ordered document lists. There is no possibility to use semantic metadata in an adequate way.

Advances in computer graphics hardware and software offer new possibilities not only for computer games but also for daily-use applications. Information visualization techniques, together with the ability to generate realistic real-time, interactive applications can be leveraged in order to create a new generation of document explorers. An application environment that resembles real environments can be used more intuitively by persons without any prior computer knowledge. The users perceive more information about the documents, such as

their size, location, and relation to other documents, visually using their natural capabilities to remember spatial layout and to navigate in 3-D environments, which moreover frees cognitive capacities by shifting part of the information-finding load to the visual system. It is hoped that this leads to a more efficient combination of human and computer capabilities: the computer has the ability to quickly search through documents, compute similarities, calculate and render document layouts, and provide other tools not available in paper document archives, while humans can visually perceive irregularities and intuitively interact with 3-D environments. Last but not least, a well-designed virtual-reality-like graphical document explorer is more fun to the average user than a conventional one and thus more motivating.

This work seeks to evaluate existing 3D file system and document collection approaches and, using these experiences together with new ideas and techniques, to develop and implement a stereoscopic and immersive system for efficiently organizing documents. Special emphasis will be on navigation in information spaces, which includes, for example, the visualization of relations between documents and the possible ways of somehow "moving" from one document to another—i.e., the realization of a seamless 3-D user interface to the semantic desktop.

2 State of the Art

2.1 The Definition of Information Visualization

Visualization is the mapping of abstract data to visually perceptible attributes like color, shape, size, location, texture, etc. with the intention of leveraging the visual system to recognize or view patterns, trends, irregularities, and connections that are not easily perceivable in text-only data collections. This can be done because evolution equipped humans with the ability to think, abstract, remember, understand, and organize visually. Thus, using these visual capabilities is completely natural, ergonomic, and efficient for human beings.

Scientific visualization focuses on the visualization of data obtained from experiments or simulation of natural or technical phenomena. This data is mainly of the numerical type and the values often have an intrinsic spatial representation. *Information visualization*, by contrast, is concerned with the visualization of *any* kind of data, from graphs to film to tables to web pages. Since information visualization plays a central role in this work, an overview will be given in the following subsections.

As Tufte put it, the goal of information visualization is the display of as much data as possible in an area as small as possible in a way such that the perception of information per time is maximized. The intention of a visualization might be the presentation of information, the "examinination" analysis of hypothetical assumptions, or the "exploration" analysis of yet-unknown datasets. The data to be visualized can have the form of tables, graphs, hierarchies, multimedia data, processes, hybrid data, or any other form. In order to accomplish with this

variety of data, a huge diversity of information visualization methods has been developed.

2.2 Visualizations and Navigational Concepts for Document Information Spaces

There are many examples of visualizations of document information spaces in the literature, and many concrete applications presented on the Web. This section aims to give a short overview of various approaches.

The *information cube* introduced by Rekimoto et al. [1] can be used as shown in Fig. 1 to visualize a file system hierarchy. The nested-box metaphor is a natural way of representing containment. Rendering the boxes semitransparently allows the user to gain an overview of the structure. One problem with this approach is the difficulty in gaining a global overview of the structure, since boxes contained in more than three parent boxes or placed behind boxes of the same tree level are hard to observe. The authors also suggest using transparency to visualize groups like aggregated nodes in a 3-D graph by rendering semitransparent spheres containing the aggregated nodes.

Fig. 1. Rekimoto et al.'s InfoCube

Card et al. [2] present some metaphors for the visualization of collections of web pages. The web pages resulting from, for example, a search query or from a person's home page, are visualized as a so-called *WebBook*, which is labeled and allows page-turning and the ruffling of pages (both animated) as in real books. The authors claim that the familiar book metaphor with its various features allows intuitive use. WebBook can be zoomed in and out to different degrees, as with a magnifying glass. WebBook is integrated in an 3-D environment called *Web Forager* (see Fig. 2). In this 3-D space, web pages and WebBooks can be placed in different distinct locations like a desk, a focus plane, the air, or a bookcase.

Fig. 2. The Web Forager

Taking into account the cost structure of information workspaces and the locality of reference principle (which states that users tend to interact repeatedly with a small number of documents), Card et al. decided to create a *hierarchical workspace* with three main levels that can be used with increasing cost of time.

The first level is the focus plane, where one page or a book can be shown. The second level is called "immediate memory space". In this level the documents can be placed on the desk or at several distinct z-planes in the air. The third level is a bookcase, placed parallel to the z-axis in 3-D space, to store WebBooks. These books can be moved to other places in space and opened, but there is also the possibility for the user to move in an animated way to face the bookcase.

WebForager's hierarchical workspace is certainly a useful way of organizing and navigating documents in 3-D space. One drawback of this system, however, is that the user gets (apart from the possibility of one search query) no computer assistance in organizing the documents in space as well as in mental categories.

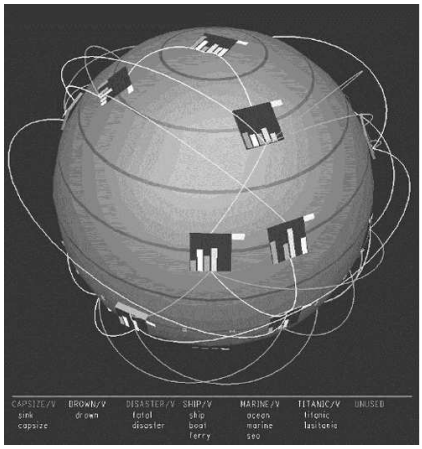


Fig. 3. The 3D NIRVE system

3D NIRVE, the 3-D information visualization presented by Sebrechts et al. [3] and shown in Fig. 3, helps the user to organize documents that result from a previous search query. Concepts consisting of one or more keywords are named and color-coded in the bottom of the window. Documents that contain all keywords of a concept are grouped together in a cluster and placed on the surface of a sphere. The more concepts are found in a document, the closer it will be to the north pole. In general, documents on the same latitude of the sphere all have the same number of concepts. The most important features of this system, with respect to user behaviors, are the grouping of documents into clusters and the color-coding of concepts.

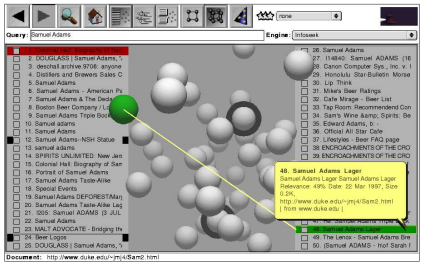


Fig. 4. The Lighthouse system

The *Lighthouse* system described by Leuski et al. [4] is used to visualize search results of web queries by integrating the traditional ranked list with a clustering visualization; the user thus benefits from the advantages of both approaches. Figure 4 shows how the ranked 2-D text lists of the results are connected to the visualization of the spheres representing the web pages. The cluster visualization places the document-spheres according to statistically determined similarities (similar documents are drawn close to each other). Since closely associated documents tend to be relevant to the same requests (cluster hypothesis), the cluster visualization gives the user more possibilities to find what she or he is looking for than a ranked list.

There are also some extra features to let the user judge the relevance of some of the documents and to use this information to calculate the relevance of unjudged documents. When in suggestion mode, this relevance value is visualized by shades of red and green, plus stars to mark the potentially most relevant documents.

A disadvantage of this system is that the double representation of documents forces the eye to often look to and fro, and maybe even to adapt the focus, when the sphere is positioned in the distance.

Robertson et al. [5] developed *Task Gallery*, a 3-D window manager, that can be regarded as a simple conversion of the conventional 2-D windows and desktop metaphors to 3-D. The metaphor used for this approach is an art gallery—a linear hallway with a platform at the end. The current task is placed on the platform, and an ordered and an unordered stack help organizing the windows. The user carries a so-called "start palette" in his or her virtual hand. It looks like an artist's pallette (in line with the gallery metaphor) and can be used exactly in the same way as the MS-Windows Start Menu to start applications or to open documents. Inactive tasks are represented by screenshots of their window configuration when they were on the platform, placed like artwork on the walls or even on the floor or ceiling of the gallery. By clicking on them, an animation is started to move the current task to its place on the wall and to move the selected task to the platform, making it the new active task.

Fig. 5. The Task Gallery

Figure 6 shows a screenshot of the *Tactile 3D* [6] system, a commercial 3-D user interface for the exploration and organization of documents; it is still in development. The file system tree structure is visualized in 3-D space using semitransparent spheres that represent folders and that contain documents and other folders. The attributes of documents are at least partly visualized: the file type defines the shape and the texture of the 3-D object representing the file. Special folders like drives also have special shapes. Hidden or read-only status is indicated by semitransparent material or by a red hue, respectively. The rotation speed of the objects informs the user about the last modification date of the object.

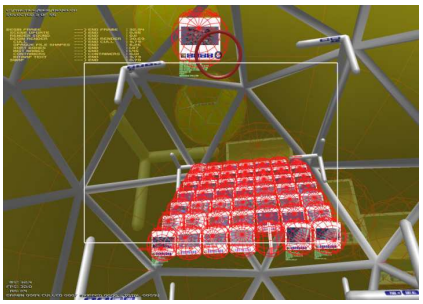


Fig. 6. Tactile 3-D

Objects within a container can be placed in a sorting box and thus be sorted by various sorting keys in the conventional way, forming 3-D configurations like a double helix, pyramid, or cylinder. The objects that are not in the sorting box can be organized by the user.

3 DocuWorld Design Decisions and Realization

3.1 The Data Visualized in DocuWorld

The data items to be visualized in the context of DocuWorld are documents with some metadata and some relations to each other. One problem in the development of DocuWorld was that not many concrete document metadata and document relations of special importance could be defined at the state of development of @VISOR, the project which subsumed this work. Therefore, some possibly useful document properties had to be defined.

Some possible attributes which might be considered as metadata of the documents, are as follows:

- (in some cases) **explicitly given data** such as the title, physical path, size (number of pages is usually more than file size), date of creation, date of last modification, date of last use, file format (e.g., PDF, HTML, plain text), document type (e.g., email, paper, report), semantic category, language, origin (author, institute), text body, and summary;
- **directly computable data** such as a thumbnail of the first page or a logo;
- **values computed by artificial intelligence** such as relevance (e.g., number of hits) according to search queries or relevance according to some example user-rated documents, important keywords, and word indexes; and
- **user defined data** such as notes, annotations, ratings, and comments.

When some of the data (for instance, number of pages) is not explicitly given, it can be computed in some cases. In the chosen document visualization, the categorical data "document category" is represented by the color of the book cover, the ordinal data "number of pages" is visualized as the thickness of the book, and a dynamic quantitative value is represented by the yellowness of the front page.

Possible types of relations could be organized in similar categories:

- **Explicitly given relations** are tree (e.g., file system tree) or graph structures.
- **Computed relations** can be, for example, similarities between documents according to search queries or other criteria.
- **User-defined relations** might be task-specific connections between documents (e.g., document A reminds the user of document B that might also be of interest to the current task).
- **Implicitly given relations** between documents are those that become clear when comparing several document attributes and recognizing similarities and differences between documents or groups of documents. These relations

are usually visualized implicitly through the visualization of document attributes.

In the case of DocuWorld, the example documents are about 200 PDF documents in the two categories "knowledge management" and "information visualization". Data for these documents includes the file name, number of pages, category, and text, including a Lucene [7] word index that allows powerful search, and is given though input methods. Other data, like relations to other documents, is artificially added to each document by dynamic initialization or defined by the user during execution time, as in the case of user-marked documents. In the final version of the application, the relations will be acquired by systems like Gnowsis [8, 9].

3.2 Global Design Decision: The Thought Wizard Metaphor

When considering the organization of documents in 3-D space, the *locomotion metaphor* "library", where the user moves through rows of bookcases, might seem a natural solution at first. There are, however, some valid arguments for the use of a *theater metaphor*, with which no movement is necessary:

- The library metaphor would work only if each document had a static position in 3-D space that could be remembered and relocated by the user with the help of his or her spatial memory. Useful information spaces should, however, allow the user to interact with the information and the visualization should react dynamically to the user's requests and needs; this is hard to realize with a static library metaphor. The concept of dynamically reorganizing document collections could be implemented with variations of the theater metaphor.
- According to the characterizing differences between locomotion and theater metaphors, locomotion metaphors are "egocentric" and theater metaphors "docucentric". In the context of a document collection application, the user's position is less important or even of no particular meaning. Therefore, the information load needed for understanding and suitably changing the user's position can be used for the more important analysis of document positions, content, and relations between documents.
- Theater metaphors also have the advantage that users with no experience in navigating in virtual 3-D environments and, in complex environments, also expert users are not in danger of getting lost in 3-D space.

The global visualization and navigation metaphor that was chosen with regard to these considerations is a variation of the theater metaphor and will henceforth be referred to as the *thought wizard metaphor*. When using this metaphor, the user (i.e., the viewpoint) is fixed and the visualization elements are dynamically moved around with respect to the current user need. Apart from restricting the viewpoint to a static point of reference, this allows the use of multiple creative dynamic visualizations: small scene elements like document representations can be moved around in the viewing volume and create various 3-D structures,

decorations or global structure elements like walls can change their appearance, and all visualization elements can be moved out of the field of view, possibly revealing other elements behind them. The mental model that should develop in the user's mind when interacting with a application implementing the thought wizard metaphor might be close to that of being an old wise woman using magic tricks to get all the information she needs, directing multiple documents simultaneously to a specified 3-D location without effort, and having the power and cleverness to do all necessary knowledge work by comfortably sitting behind her desk and using gestures and magic words.

3.3 A Short Description of DocuWorld

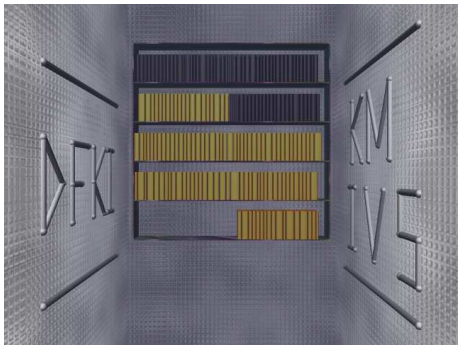


Fig. 7. Initial layout of DocuWorld

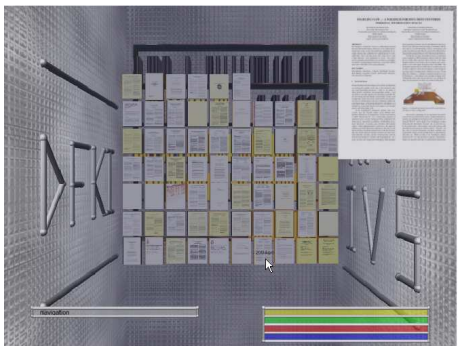


Fig. 8. Preselected documents

The DocuWorld application starts with a visualization of all documents in the state of being stored in some place in the background. As it can be seen in Fig. 7, the start layout of the final realization of DocuWorld displays all documents in the form of books standing at a bookcase in the back of the room. The user can preselect documents of this collection by typing a search query in the gray search query panel. The search panel is invoked by a gesture, which moves it to the bottom of the screen with the gray search field on the left side.

The preselected documents can be thought of as belonging to a higher semantic zoom level than those in the bookcase and should therefore be displayed with more detail. Figure 8 illustrates the visualization state of DocuWorld when some documents have been preselected. These documents are moved out of the bookcase, rotated so that their textured front page faces the user, and flown to their place in the start configuration of the plane layout (which will be described later). While the preselection search query has to be typed in the left search query panel, additional search queries typed in the color-coded right search query panels can be used for more detailed requests. The results of these search queries are visualized in various modes that allow different insights in the data collection.

Important documents will show an animated pulsing behavior and thus instantly catch the user's eye. There is also the possibility to mark documents with red crosses in order to define user- or task-specific interests and not to lose sight of them when they move to different places in different modes. To get a closer view of single documents, they can be moved to the front (the reading position). Although DocuWorld does not support these features, ruffling through pages and annotating the text could be allowed in reading mode. In addition to the reading mode, a preview image of the front page of the documents is shown in the upper right corner of the screen, when the mouse moves over documents (see Figure 8). To gain better insight in the 3-D structure of documents organized in 3-D space, the structure can be rotated by mouse gestures or rocked with an efficient predefined rotation.

3.4 Special DocuWorld Visualization Concepts

Structure on Demand The term "details on demand" is commonly used in the context of semantic zoom and interactive focus and context views. In these systems, the user starts with an overview visualization and demands more details of special visualization items. In trying to find a good compromise in the trade-off between the number of information items being displayed and the perceivability of the structure, the "details on demand" concept was transferred to the "structure on demand" concept. The idea of this concept is to find a context- and user-sensitive solution to the information structure trade-off by allowing the user to call up and remove certain structure elements depending on the current task and user specific preferences.

In the implementation of DocuWorld, the search panels can be called up and removed by a gesture, which moves them in or out in an animated way. The memory-line structure described in Sect. 3.7 and shown in Fig. 9 can also be called up on demand, which is quite useful as it is positioned in front of all other documents and thus party occludes them. Even the main DocuWorld structure, the walls, can be removed to reveal the area behind it in which 3-D structures for user-specific placement of documents are located, and called again when the walls are needed to perceive the 3-D structure of the main document visualization more easily. The bookcase and the main document visualization area can also be called up on demand.

Ghost Documents DocuWorld will allow artificially computed semantic structures as well as documents organized by users. This leads to the problem of how to treat user-placed documents: on the one hand, the mental model of the document representation is that of an object with an identity that can be located at only one place at a time; on the other hand, documents placed at user-specified locations are of special interest to the user and it might be helpful to see their position in a computed semantic structure as well, but displaying one document at different locations would contradict the idea of document identity.

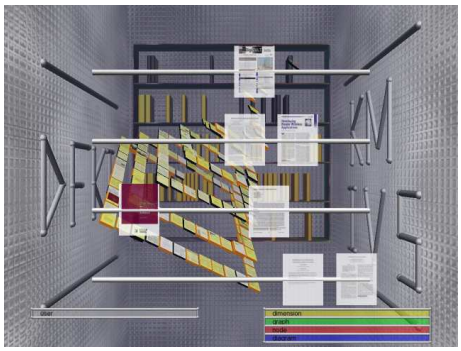


Fig. 9. Ghost documents at memory-line

The solution implemented in DocuWorld is that of *ghost documents*: "real" documents are located only in artificially organized areas. Whenever the user decides to store a document at a specific place, a "ghost" of this document is automatically created which is moved from the original document to the selected storing place, while the original document stays where it was. Figure 9 shows a computed document structure in the background and some ghost documents on the memory-line structure described in Sect. 3.7. Ghost documents, in contrast to real documents, have no 3-D shape, but only a flat, slightly transparent surface which makes them look immaterial and ghost-like. It is possible to visually indicate the document the ghost stands for with a special gesture that triggers the rendering of a semitransparent blue box around the corresponding real document.

Tactical Zoom In conventional 3-D visualizations with perspective projection, the operation of scaling and the operation of moving the viewpoint along the z-axis are isomorphic operations, which means that moving the viewpoint has the same result as a particular scaling operation. This is, however, only true for 2.5-D visualizations: as soon as stereo viewing is enabled, a scaled down object in the front can be distinguished from an object moved to the back due to binocular disparity, that is more or less obvious depending on the quality of the 3-D output device. Nevertheless, the (near) isomorphism between scaling and translating in the z-direction makes it possible to use one operation instead of the other or to combine both operations in order to create new creative visualization or interaction tools. Mine et al. [10], for instance, used automatic scaling for their *scaled world grab*: instead of moving the object to the user's hand or the user to the object, the world containing the object is scaled until the user can reach the object.

For the purposes of DocuWorld it was decided to experiment with z-axis translation and scaling in a different approach: abolishing the natural semantic connection between translation and scaling might allow use of the scale factor as an additional visualization dimension. The idea, called *tactical zoom*, is to use z-axis translation simply and intuitively to organize documents in 3-D space and to use the scale or zoom operation not as a substitute for z-axis translation, but for other, tactical purposes. In accordance with this idea, the physical size of documents would not be fixed, but rather opportunistically changed depending on the visualization situation, thus making more efficient use of 3-D space. If, for example, the documents in the bookcase need to have a particular size to

all fit into the bookcase, and another document size would be more appropriate for organizing the documents in 3-D space, they could simply be resized. If, instead of rescaling, only a conventional translation along the z-axis had been used, this would have lead to a non-optimal use of space, as in the example of the bookcase standing in the middle of the scene or even objects intersecting the borders of the viewing frustum in the case of the organic structure moved closer to the viewer. Tactical zoom is also used when objects enter or leave the reading state: as it was decided to move documents for optimal readability to the focus plane perpendicular to the z-axis because there is no binocular disparity and non-rescaled documents are still too small to be readable at that distance, they are rescaled to cover the whole screen. Of course, this also leads to conflicting depth clues. Nevertheless, it is not important to the success of the application to understand at what distance the documents in PlaneMode are compared to the one in reading mode or the ones at the bookcase. It is only essential to perceive the spacial relationships and distances between documents in one mode.

3.5 Visualization of Search Results

One DocuWorld visualization mode for search results is PlaneMode (see Fig. 10). The preselected documents are organized with the front page facing the user in an xy-plane in front of the bookcase. The size of these documents is optimized depending on the number of preselected documents according to the tactical zoom concept. Semitransparent color-coded score bars in front of the documents display the relevance of the individual search queries for each document. The translation of every document along the positive z-axis according to its search result depends on the sum of the relevance values returned by the search queries, bringing the most relevant documents to the front and thus to the focus of the user.

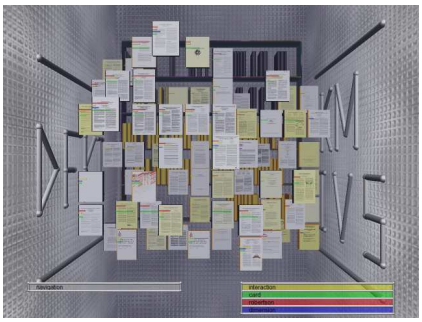


Fig. 10. Search results in PlaneMode

3.6 Document Relation Layout

The three dynamically generated relation types in the general form of "document A is related to document X, Y, ..., and Z", which create a directed graph structure, where each document is related to 0 to 25 other documents, are visualized in three different ways—one relation visualization for each relation type—which should be seen as example relation visualizations that could be exchanged or supplemented with other relations.

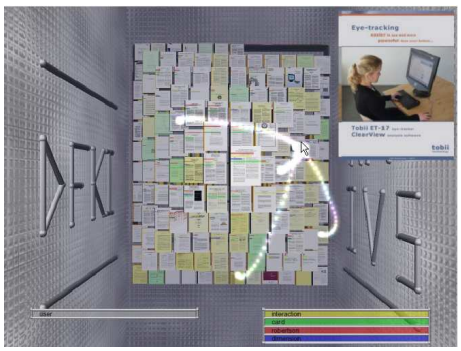


Fig. 11. Relation Types 1 and 2.

The first two relation types are visualized as shown in Fig. 11. Relation Type 1 is intuitively represented by connecting the document under the mouse pointer by curves to the documents to which it is related. These curves might create a mental model like that of *thought flashes* moving the user's attention from the current document to related documents. This mental model is suggested not only by the way the curves are rendered with shining particles, but also by the fact that the curves are animated, starting at the document under the mouse pointer and propagating to the related documents. The advantage of this way of representing relations is that documents are visually strongly connected by curves which moreover create an impressive 3-D structure. The disadvantage of literally connecting documents is that the thought flashes occlude the documents behind them, which makes it sometimes hard to understand which documents the curves are pointing to.

The visualization of Relation Type 2 tries to make up for this disadvantage: the documents which are related to the document under the mouse pointer are simply visualized with semitransparent green boxes around them. Thus, they can be perceived preattentively and do not occlude anything as the boxes are not much larger than the documents themselves. Also, the box representation can be used without complications in PlaneMode and in the bookcase. Compared to Relation 1, the connection of the document under the mouse pointer and the ones in green boxes are, however, not as intuitively clear as in the case of connecting thought flashes. One possible solution to this dilemma might be to combine the visualization of both Relation Types and their advantages in the visualization of a single relation type—i.e., to use thought flashes that end at documents in green boxes.

The visualization and use of Relation Type 3 is different from the previous ones. When enabled, the user has to trigger the display of the relation by clicking on a document. This will display the documents the selected document is related to in the form of ghost documents hovering before the main document space. The user now has the possibility to select one of these ghosts by clicking on it, which will trigger an animation that seems to collapse all of the ghost documents to the real document the selected ghost stands for, and to further visually emphasize this document by displaying a semitransparent red box around it in the moment of collapse. With this visualization technique for relations the user's attention is automatically moved from the selected document to all related documents and finally to the related document that seems to be most useful to him or her. From

this document he or she might start the process again to move one step further to related documents.

3.7 Documents at User Defined Locations

The previously described document collection visualization modes are for the artificially computed organization of documents. To give the user the possibility to organize documents according to his or her own organization principles and to remind him or her of previous thoughts, it was decided to also support user-specific structures. Section 3.4 introduced ghost documents as a way of side-by-side use of user structures and computed structures without confusing double representations of documents.

DocuWorld provides two different areas for user-defined document placement: the memory-line in front of all other structure elements which can be seen in Fig. 9, and the area behind the environment walls where ghost documents can be placed on human statues as illustrated in Fig. 12, in which the wall structure is at an intermediate state of the animation of being lifted to an invisible place. The memory-line, which was derived from the idea of a clothesline, allows the user to hang about twenty important or frequently accessed documents on it. In the area behind the walls can be placed more ghost documents that are less frequently accessed. Thus, DocuWorld provides a hierarchical workspace similar to Card's [2] WebForager presented in Sect. 2.2. The choice of human figures as a storing place was made because human figures, small changes in their appearance, and thus also objects stored on human bodies are easy to recognize and to remember—much easier than remembering the location of documents on a conventional bookcase. The current DocuWorld implementation uses only two human statues at fixed locations. This concept could, however, be extended to add (differently shaped) statues when the user requests it, to drag and drop them to any place behind the environment walls, and to move them to the center of the visualization area and rotate them.

Fig. 12. Statues as storing places

4 Evaluation and Conclusion

In this work existing document collection systems were evaluated and DocuWorld, a prototype for a new immersive 3-D system was developed.

The difficulties of moving from 2-D interfaces to 3-D interfaces were identified and solved: it was explained in detail how the layout of 3-D visualizations should be optimized to make good use of the available 3-D space and the Thought Wizard metaphor was developed as an efficient and flexible global visualization and navigation concept that dynamically moves the visualization items to the user instead of forcing the user to orient him- or herself in a 3-D environment. The need for two types of document areas, one for user-specific document management and one for automatically computed document organizations, was established, memory-line and human statues were developed as ergonomic user storing areas, and as a solution to the problem of double representations of documents, immaterial ghost documents were used.

Compared to the *Web Forager* approach presented in section 2.2 DocuWorld provides the user with a more natural environment, more flexible possibilities to store documents at user defined locations, and the visualization of document semantics through automatic document organization modes and visually indicated document relations — a feature not available in the *Web Forager* approach.

All these aspects, together with the visualization of the document metadata, search results, and the relations between documents, allow preattentive processing of the data, easy and intuitive use, immersion in the animated 3-D environment, and navigation from one document to the other by simply using the Thought Wizard tools or displaying document relations. Although still in a prototype state, DocuWorld is an example realization of a 3-D document collection system that was tested with research lab staff. During these tests the intuitive and animated visualization proved to be easy to use and more enjoyable than a conventional document explorer although there still are many restrictions in the provided functionality. Moreover, the visual representation of document semantics as the relevance to search queries and relations between documents allow the user to quickly perceive relevant documents and to navigate from one document along the displayed relation to a related document. These possibilities of representing document semantics can hardly be provided by traditional 2D text based interfaces, which indicates that a 3D user interface like DocuWorld is the appropriate way of implementing the Semantic Desktop.

To move the presented approach from prototype state to a usable application, additional research and implementation efforts will be necessary. Future work will, for example, have to focus on the replacement of keyboard gestures with more natural, easy-to-remember, and immersive (hand or body) gestures. The 3-D graphics could also be improved to create more convincing and impressive effects. Furthermore, the Gnowsis framework will be integrated in DocuWorld to acquire relations and other semantic metadata. Moreover, supplementary possibilities of information retrieval and knowledge acquisition as well as possibilities to interact with document collections like dragging documents through space or saving interesting configurations and document arrangements for later use will have to be added to create a powerful application that does not restrict the user.

References

1. Rekimoto, J., Green, M.: The information cube: Using transparency in 3d information visualization. In: Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93). (1993) 125–132
2. Card, S.K., Robertson, G.G., York, W.: The webbook and the web forager: an information workspace for the world-wide web. In: CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (1996) 111–ff.
3. Sebrechts, M.M., Cugini, J.V., Laskowski, S.J., Vasilakis, J., Miller, M.S.: Visualization of search results: a comparative evaluation of text, 2d, and 3d interfaces. In: SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (1999) 3–10
4. Leuski, A., Allan, J.: Lighthouse: Showing the way to relevant information. In: INFOVIS. (2000) 125–130
5. Robertson, G.G., van Dantzich, M., Robbins, D.C., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., Gorokhovsky, V.: The task gallery: a 3d window manager. In: CHI. (2000) 494–501
6. : (Tactile 3D User Interface: http://tactile3d.com/)
7. Lucene: (http://lucene.apache.org/)
8. Sauermann, L.: The gnowsis — using semantic web technologies to build a semantic desktop. Master's thesis, Technical University of Vienna (2003)
9. Sauermann, L.: The semantic desktop - a basis for personal knowledge management. (2005) 294–301
10. Mine, M.R., Brooks, Jr., F.P., Sequin, C.H.: Moving objects in space: Exploiting proprioception in virtual-environment interaction. Computer Graphics **31** (1997) 19–26

Human and Social Aspects of Decentralized Knowledge Communities

Indratmo and Julita Vassileva

Department of Computer Science, University of Saskatchewan
Saskatoon, SK S7N 5C9, Canada
j.indratmo@usask.ca, jiv@cs.usask.ca

Abstract. To design an infrastructure for knowledge communities, we need both technical expertise and an understanding of human and social aspects of communities. Technologies for implementing such infrastructures are often available. However, there is no clear, proven procedure for building successful communities. In this paper, we review research literature concerning user practices and social aspects of information and knowledge management. Based on this review, we propose preliminary design criteria for Semantic Desktop systems.

1 Introduction

Designing an infrastructure for knowledge communities requires not only technical expertise, but also a proper understanding of the human and social aspects of communities. Understanding these aspects is necessary because the success of communities often depends on subtle design issues [1, 2].

The structure and dynamics of a community emerge from local interactions of its members. These interactions cannot be controlled to produce specific collective behaviour, as each community member is autonomous and interdependent. However, we can support communities by designing an infrastructure that fosters characteristics of successful communities, such as a high level of participation, contribution, and cooperation among community members.

Recently, a vision [3] has emerged of building peer-to-peer (P2P), semantically rich knowledge communities. The basic idea is to apply Semantic Web principles [4] to personal information management (PIM), resulting in Semantic Desktop systems (e.g., [5, 6]). Then these systems are interconnected using a P2P protocol, providing an infrastructure for online communities. Some of the technical aspects of such infrastructures have been discussed elsewhere (e.g., [3, 5–7]). However, the discussions so far have overlooked the existing user practices in PIM and social aspects of communities. Unless the new infrastructures accommodate these practices, they will not be adopted by users, and their applications will be severely limited.

In this paper, we review previous research on personal information management and social aspects of knowledge management, so that lessons from many user studies about PIM and online communities can be used for implementing the

vision of P2P knowledge communities [3]. The paper consists of five main parts: (1) user practices in PIM, (2) problems in information management, (3) various approaches to solving the problems, (4) design criteria for Semantic Desktop systems, and (5) social aspects of knowledge communities.

In the first three parts, we give background information about PIM—activities in organizing and managing information for personal use. The discussion focuses on human factors in PIM, which then become the basis for the development of criteria for designing Semantic Desktop systems. In the last part, we discuss social aspects of communities, such as the role of social networks in information and knowledge management, and factors that motivate users to contribute to their communities.

2 User Practices in Personal Information Management

This section discusses existing research on user practices in personal information management. First, it outlines common purposes of PIM, then discusses different types of filing behaviours, and finally highlights the importance of various cues in information management.

2.1 Basic Purposes: Finding and Reminding

People manage their documents not only to make it easier to find them later, but also to remind themselves of their tasks [8, 9]. To make finding documents easier, people arrange the documents according to some logical order. To achieve the reminding function, they make relevant documents more visible than others. For example, putting a document on a computer desktop can remind the user to work on the document.

To support finding and reminding, people organize their documents using "files" and "piles" [8, 10]. Files are organized document collections and usually contain archived information. Piles, on the other hand, are disorganized document collections and typically contain work-in-progress documents. A pile can function as a reminder and a temporary organizational unit. Although not organized explicitly, elements of a pile may naturally follow reverse chronological order [8].

When searching for specific information on their personal computer, users prefer to browse their directories manually than to use a search tool [9, 11, 12]. This preference may result from users' familiarity with their personal workspace: they feel that they know where they have saved a document. Users tend to use a search tool only when they cannot find a document manually, or when they search information that is available outside their personal workspace, such as on a shared file server or on the Web.

2.2 Filing Behaviour

A common activity in information management is to file documents; that is, to place documents in an appropriate folder or category.

People's behaviour in organizing their documents ranges from "pilers" to "filers" [12–14]. Pilers are users who do not categorize their documents. An example is an email user who leaves all incoming messages in the inbox. Filers, on the other hand, are those who use folders extensively; they create many folders and classify their documents into various folders on a daily basis.

Personality traits and job types affect people's filing behaviour. Some people tend to be more organized than others. People whose tasks are structured and routine tend to file their documents more frequently than those whose tasks are unstructured [8]. For knowledge workers [15], for example, filing information is often not so important, as their work relies heavily on creativity and knowledge that is in their mind. Thus, knowledge workers focus on absorbing information contained in documents, rather than relying on them.

Researchers have observed filing behaviour in various applications, such as email [12, 14], Web bookmarks [12, 13], and file systems [12]. The classification of filing behaviour in these studies, however, is not absolute. People's behaviour is complex, so it is possible to refine the suggested classification schemes. Boardman and Sasse [12] observe that users in practice use multiple strategies to manage emails and Web bookmarks. Specifically, some email users file important messages immediately (frequent filers), but leave other messages in the inbox and organize them occasionally (spring cleaners). Similarly, Internet users organize some of their new Web bookmarks at the creation time (creation-time filers), but leave others as they are (no-filers).

The research discussed above studies how frequently and when people file their documents, but it does not discuss how people group their documents. What factors affect classificatory decisions?

Documents' intended use or purpose appears to be a strong factor that affects classificatory decisions [16]. People tend to group documents based on the task related to the documents, such as grouping all materials for teaching a course Computer Science 101. Basic properties of documents—their size, type, or author—are less influential than their intended use. In organizing photos [17], however, people typically use temporal information, such as events or date, as the main factor in making classificatory decisions.

2.3 The Importance of Context and Cues

People use various cues to maintain contextual information of their documents. A business card attached on a paper may serve as a reminder to send the paper to a particular person. A date written on top of a document may remind the user of a project's deadline. A red folder may indicate the urgency of documents within it. These cues, however, can be interpreted and used in different ways. Therefore, knowing the context in which such cues are being used is crucial.

Context is important for a proper understanding of information and in the recall process [18]. A classic example, as discussed in [18], is a chess study [19]. This study suggests that chess experts can reconstruct chess positions from real play well because the positions are meaningful to them, which helps in recall. When the positions are randomized, their ability to recall declines significantly.

In the context of information management, associating a document with various cues—visual, spatial, chronological, or contextual—can improve people's ability to find the document because humans are good at remembering the appearance, essence, and context of a document. Without such cues, people have to remember details, such as filenames, which they are not good at.

Spatial locations of documents also carry semantic information to the user. In office organization, people keep documents that are more recent or urgent closer to the centre of their working area [8, 20]. In desktop organization, computer users arrange their screen systematically [11]: they group program shortcuts on a specific region, put related documents close to each other, and try to arrange their screen's layout symmetrically. Such spatial organizations can help users locate frequently used programs and documents on their computer.

3 Problems in Information Management

Current systems for PIM are far from being perfect. Studies report four areas where improvement is necessary: supporting information organization better, maintaining context and retrieval cues, dealing with the problem of information overload, and supporting interoperability between individuals or organizations.

3.1 Filing

Filing information is difficult. It has been observed in various studies, including office organization [8, 10], email management [14], and desktop organization [11]. In these studies, users state that filing is a heavyweight cognitive activity: they cannot make classificatory decisions easily. When filing a document, users try to put the document into a category in which they expect to remember it for later retrieval. Because there are many ways to organize documents, as discussed in section 2.2, a classification scheme that is currently appropriate may become unsuitable as the user requirements or goals change over time.

The currently predominant hierarchical structure in information management makes filing information difficult and inflexible. In the physical world, people are limited to having to put a document into one specific folder, unless they make a copy of the document, which then can be put in a different folder. Unfortunately, most file systems and computer applications impose the same restriction: they do not allow users to classify a document into multiple categories (e.g., folders or directories). Although users can alleviate this problem by making copies of documents or links to files, they normally do not do this [10].

Hierarchies also restrict users from capturing multiple semantics of a document. Where should a user save a paper about the Semantic Web and ontology, in a "Semantic Web" or an "Ontology" directory? Filing the paper in either of these categories causes loss of information about the document's content. And it complicates the retrieval process, as the user has to remember exactly the location of the paper. Although the user could create a "Semantic Web and Ontology" directory, it is uncommon to have a category with multiple semantics.

The ambiguity of natural languages also contributes to the difficulty of filing information. A word can refer to different things. A category "Networks" may refer to network protocols, Internet applications, or even network marketing. Thus, it is difficult to choose good names for semantic categories. In addition, it is hard later to remember the intended semantics of the categories, as the mind-set of the user may change in the meantime. Often a category created with a specific semantics "evolves" due to filing of documents that do not exactly fit in it, because users are reluctant to create a new category, since this may require reorganization of the previously classified documents.

3.2 Maintaining Context and Retrieval Cues

Computer systems offer limited supports for maintaining context and retrieval cues in information management. Visual, contextual, and spatial cues are important for finding and reminding [18]. In the physical world, tangible documents have rich features, such as colour, size and thickness, which can serve as retrieval cues. For example, when searching for a book, people may forget the book's title, but they usually can still remember the book's appearance. Therefore, they can limit their search based on these cues. Unfortunately, many of these cues are lost in the digital world. Instead of maintaining visual retrieval cues, computer systems display details of files, such as their name, type, and size, which are less useful for retrieval cues. Furthermore, there are only limited possibilities for users to arrange the spatial layout of their document collections.

Filing documents removes some contextual information of the documents. Suppose that in an office, a user keeps urgent documents in a red folder. While the folder is on a desk, it can remind the user of the urgency of the documents within it. However, when the user puts the folder into a filing cabinet, the user can no longer see the urgency denoted by the red colour. A similar effect applies to digital documents. For example, saving an email attachment in a file folder removes the contextual information (e.g., from whom and when the document was received), which is usually available in the email body [20].

Context can also be lost because of some habitual use of computer applications. For example, an email's subject and its content are not necessarily consistent, as users often simply press "reply" to an old message when starting new conversational threads. In a long conversational thread, users may not include all previous messages when sending a reply. They may also discuss topics that are irrelevant to the initial topic of discussion without changing the email subject.

3.3 Information Overload

People often have to deal with amounts of information exceeding their processing capability. This phenomenon appears in various studies of management of files [9], paper archives [21], email [14], and social networks [22, 23]. Across these studies, the main problem is related to the volume of information that people have to manage. On a closer look, however, there are finer issues in the information overload problem as discussed below.

People need to manage many ephemeral documents [9], such as emails and memos. Ephemeral documents usually serve as reminders, so they should be kept visible. However, computer screens are spatially limited, so users cannot arrange such documents freely. Moreover, the lifetime of these documents sometimes depends on other people. For example, a note is only relevant until a reply from another person arrives. Unfortunately, not all people respond to others' requests timely, making the management of ephemeral documents more difficult.

Studies [14, 24] observe that people use email not only for communication, but also for reminders, personal archiving, and task and contact management. These practices exacerbate the information overload problem, as they cause users to receive and manage more messages without using the right tool.

Another problem related to information overload, called premature filing, is observed in the management of paper archives [21]. This problem is described as follows. When receiving many documents, people cannot always decide the usefulness of these documents immediately. While they do not want to keep useless documents, they consider the potential value of these documents and hence hesitate to discard them right away. So they decide to file these documents, hoping to find time to assess their value later. Unfortunately, filing makes documents less visible. As people receive many new documents, they often forget about their filed documents. As a result, they keep many documents, which have little value to them, and are often discarded later without ever being read [21].

3.4 Interoperability

Problems in personal information management escalate when people have to share documents. People are familiar with their personal workspace: they organize it personally, know the contents of their document collections well, and decide the semantic organization of their workspace by themselves. These characteristics, however, do not exist in a shared repository: several people have access to the repository; others may add new documents to it; and as a team, they have to agree to some convention of how to organize this repository.

Due to the rich semantics in natural languages, people often use different words to refer to the same concept, and use the same word to refer to different concepts. According to Furnas et al. [25], the probability of two persons choosing the same word to describe the same concept is less than 0.20. In other words, it is unlikely that people will use the same vocabulary to describe the same things.

In collaborative work involving different groups of people, each group may require local customizations of documents, have different views of the shared objects, or favour a certain filing scheme [26, 27]. Dourish et al. [27] report that in an organization that has a standard filing scheme, each group still needs to adjust the standard filing scheme to meet requirements of different projects. As each group has its own local view, document sharing between groups becomes complicated. To be understandable, local customizations should be presented to each group differently, ideally, according to the corresponding group's local view.

Even for a small, relatively homogeneous team [28], interoperability problems exist. These problems include personal preferences over coarse- or fine-

grained categorization, topic- or purpose-based categorization, and syntactic- or semantic-based categorization.

4 Approaches to Handling Problems in Information Management

A variety of approaches have been proposed to deal with the problems discussed in the previous section. We focus on four main approaches below: tagging, flexible collections, temporal-based organizations, and the Semantic Desktop.

4.1 Tagging

The basic principle of tagging is to assign attributes (i.e., ⟨field, value⟩ pairs) to documents, and then to allow users to use these attributes to organize and retrieve documents [29–31]. This principle enables users to assign multiple attributes to a document, so they can express the semantics of a document more flexibly. Users are no longer limited to having to classify a document into a single category as with filing.

The concept of tagging has been applied to several file systems [29, 30, 32]. These systems use tagging for providing a flexible way of retrieving and grouping files. To retrieve and group files, users are no longer restricted to one strict hierarchical structure. Instead, they can group files based on a certain attribute of the files. For example, in a Semantic File System [29], to retrieve all papers about Semantic Web, a user can submit a query "topic=semantic web"[1] and the system will create a virtual directory containing all files that match the query.

Attributes have been used not only for providing associative access in file systems, but also for providing personalized services [33] and managing documents [31, 34, 35]. In general, these systems use attributes to capture metadata and to provide a flexible way of organizing and retrieving documents. Placeless Document Systems [34], however, extend this usage by allowing an attribute's value to be active code. So, for example, users can assign a program to a document, which backups the document periodically. This allows to associate not just semantics, but also computation with the corresponding documents.

4.2 Flexible Collections

Collections are an abstraction for handling the limitations of a hierarchical structure and for providing a flexible way of organizing various types of documents. There are two main principles of collections [31, 36]: (1) a document can belong to multiple collections, and (2) a collection can contain different types of documents.

The first principle allows a document to become a member of several different collections. This membership can be static, dynamic, or a combination of static

[1] The actual form of queries in Semantic File Systems is specified as virtual directory names, e.g., /sfs/topic:/semanticweb.

and dynamic. For example, semantic directories [30] and fluid collections [31] enable users to specify an inclusion list, an exclusion list, and a query to create a collection. The inclusion list contains a list of documents that must be included in the collection regardless of the query. The exclusion list specifies documents that must be excluded from the collection even if they match the query. The query states the criteria of documents to be included in the collection.

The second principle enables users to put various types of documents—such as emails, Web bookmarks, appointments, contact lists, and Word documents—together into a collection. Users can organize their documents in a more logical and meaningful way, as document organization is abstracted from the applications that produce the documents.

4.3 Time-Based Approaches

Time-based approaches provide an alternative to the desktop metaphor for managing information. These approaches are based on the fact that time is an important retrieval cue in information management [8, 17, 37]. Examples of such approaches include Lifestreams [38] and Time-Machine Computing [39].

Lifestreams [38] stores a user's personal information chronologically as a *lifestream*. A lifestream contains old documents, working documents, and possibly, future documents, such as appointments and reminders. A document's name is not mandatory because the system is responsible for identifying and placing the document in the lifestream. To facilitate information organization and retrieval, the user can create substreams to display only subsets of documents in the lifestream. Within a substream, the user can refine the query further by creating other substreams. Finally, Lifestreams is also capable of summarizing information in a substream and presenting the result to the user.

Time-Machine Computing (TMC) [39] extends the idea of Lifestreams [38] by capturing both temporal and spatial information of a document. To do so, TMC provides a special desktop that allows users to keep and organize their documents spatially on this desktop. Users may remove documents from their desktop. However, these documents are not deleted permanently because TMC keeps track of any state changes on the desktop. Thus, users can access their removed document by setting their desktop's time back to a point before they remove the desired document, and TMC will restore the state of the users' desktop at the specified time.

4.4 The Semantic Desktop

The core technology of the Semantic Desktop is the Semantic Web [4]: an extension of the current World Wide Web that enables machines to "understand" and process information intelligently. As discussed in [4], the main components of the Semantic Web consist of the Extensible Markup Language (XML), the Resource Description Framework (RDF), and ontologies. XML enables users to create arbitrary tags for describing the structure of documents. RDF provides a

framework to express the meaning of information using XML. And an ontology—"an explicit specification of conceptualization" [40]—makes information sharing meaningful by providing a set of vocabulary to discuss a particular domain.

The main idea of the Semantic Desktop [3, 5, 6] is to apply principles of the Semantic Web [4] to personal information management. This approach allows the creation of semantically rich PIM tools. Semantic Desktop systems can use RDF to express both the structure and semantics of webs of information on a user's personal computer as metadata. Because this metadata can be "understood" by machines, this approach has potential for improving current practices in PIM: information overload can be reduced by delegating well-defined tasks to agents; information sharing can be more meaningful and contextually rich, as the corresponding metadata is also shared and described using common ontologies.

Like the Semantic Web, the Semantic Desktop treats information as a Web resource, so information is identified using a Uniform Resource Identifier [3, 6]. Some projects developing Semantic Desktop systems include Haystack [5], Gnowsis [6], Chandler,[2] and Fenfire.[3]

5 Design Criteria for Semantic Desktop Systems

To be adopted widely by users, Semantic Desktop systems must be designed properly. One of the most important aspects in design is to know the users. Thus, based on the existing user practices and limitations of approaches to personal information management, we propose preliminary design criteria for evaluating Semantic Desktop systems as follows:

- **Flexibility:** A Semantic Desktop system should support flexible information organization. This organization should be abstracted from both the structure of information storage (e.g., hierarchical file systems) and desktop applications (e.g., email tools, Web browsers, and task management tools). Grouping and regrouping information should be easy. Users should be able to arrange information based on their logical views, intentions, or usage patterns. For example, users should have options to sort information based on its frequency of access, its level of importance, or its level of urgency. Flexibility is important because users have different needs and preferences in organizing information (see section 2.2).
- **Retrievability:** Users should be able to find desired information easily. Just like other Web resources, users' personal information should be accessible virtually from anywhere. Users do not need to know where their data actually resides on. And since users like browsing their personal workspace [9, 11, 12], a Semantic Desktop system should provide a good browser, which supports multiple information visualizations and allows users to see cues and contextual information of their documents to facilitate retrieval. In other words, the search mechanism should favour recognition over recall.

[2] http://www.osafoundation.org/Chandler_Compelling_Vision.htm
[3] http://fenfire.org/vision.html

- **Security:** While easy access to information is desirable, security must not be compromised. Every access to the user's data must be authorized. Sensitive information should be transferred through a secure channel and encrypted. A Semantic Desktop system should authenticate shared data and metadata. Also, it should allow users to control access privileges of their information (e.g., public, group, or private data).
- **Context maintenance:** Contextual information of documents is usually stored as metadata. Creating metadata sometimes requires manual input from users. Unfortunately, users are generally reluctant to do more work, such as typing additional information about a document. Thus, a Semantic Desktop system should be able to capture or create metadata automatically, for example, by using context analysis [41] or data-mining techniques. If user input is unavoidable, then the system should provide a mechanism that allows easy annotation (e.g., [42]). Presenting summaries of interactions [43] can also help users recall the context of interactions while communicating with other people.
- **Proactiveness:** A Semantic Desktop system should be proactive in promoting awareness to the user. Examples include notifying users when they receive emails from their important contacts, highlighting important or related documents, and reminding users of their appointments and tasks.
- **Cognitive load:** A Semantic Desktop system should minimize the user's cognitive effort necessary to use the system. Reducing cognitive load can be achieved using various ways, such as encapsulating technical details from users, enabling users to delegate well-defined tasks to agents, presenting summaries of information, and filtering in/out incoming information.
- **Interoperability:** Interoperability can be examined from different perspectives. At a system level, a Semantic Desktop system should promote interoperability both among desktop applications on a single computer and with other Semantic Desktop systems. At a user level, the system should resolve inconsistencies among different ontologies used by users.
- **Performance:** A system performance should be reasonable and acceptable to users. Responses to user queries should be timely. Resource consumption and allocation should be well managed.

Evaluating a PIM tool rigorously requires a well-designed user study and in-depth analysis. While such rigorous analysis is out of scope of this paper, we present simple analyses to illustrate trade-offs of existing approaches to PIM.

Most operating systems and desktop applications currently offer hierarchical structures for information organization. Such structures are not *flexible* enough to meet the user requirements (see section 3.1). Consequently, users feel that organizing information is a heavyweight *cognitive* activity. This inflexibility also complicates information *retrieval*, as the *context* of documents can only be maintained in a limited way. Despite their disadvantages, hierarchical tree structures are computationally efficient and hence deliver good *performance*.

The Semantic Desktop views information, regardless of its type, as a Web resource. This view allows *flexible* information organization and easy *retrieval*.

This approach promotes a *contextually* rich PIM environment where metadata is described using RDF and ontologies, which support *interoperability*[4] among applications and with other systems. *Proactive* agents can be used to deal with information overload. The Semantic Desktop, however, is complex, and therefore should minimize the user's *cognitive* load so that it is usable by average users.

6 Social Aspects of Knowledge Communities

We have reviewed practices, problems, and approaches in information management from individual users' point of view. While this review is useful for developing a PIM tool, to build successful knowledge communities, we should also consider social aspects of communities. What is the role of social networks in information and knowledge management? How can we use social networks to develop a better infrastructure for knowledge management? How can users be motivated to participate and contribute to their communities? These issues are discussed in this section.

6.1 The Role of Social Networks

Social networks [44] study relationships and information flows among people, and play an important role in information and knowledge management. People set a high priority to messages that they receive from families, bosses, or closed friends. They prefer to collaborate with their trusted friends. And they are more willing to share their knowledge in an environment where they feel accepted and secured—safe to speak up their minds, safe to make mistakes, and so on. In other words, trust and interpersonal relationships are important. It is not surprising that people make considerable efforts in maintaining and expanding their social networks [22].

Informal conversations, which usually happen in office lounges or other recreational places, have potential for building trust and relationships [45]. In such relaxing places, people can talk informally about work or personal things while developing relationships with other co-workers. In computer systems, such informal conversations can be supported by chat applications [45, 46].

The main activities in maintaining social networks include remembering and communicating [22]. Remembering people's expertise is necessary for locating the right resources. Remembering personal details enables people to give a personal touch to their relationships, for example, by sending birthday or anniversary cards. To facilitate remembering, people maintain intermittent communications, especially with members of their social networks who are currently inactive, such as with collaborators from past projects. Communication is helpful to refresh memories of both parties and to maintain connections between them.

[4] Achieving large scale interoperability, however, is a non-trivial task.

6.2 User Motivation

Communities need a critical mass of users. Not only the number of users is crucial, but also their participation in the communities. Without a critical mass, the systems underlying communities will be abandoned by the users [47, 48].

Every person has a personal goal and motivation to join and participate in a knowledge sharing community. Some may want to meet new people with similar interests while others want to learn and follow the development of a research area. Although individual goals may vary, members of a community usually share a common interest to some degree. In general, people will stay active in a community if by doing so they can achieve their goals. Despite the variety of individual goals and motivation, how can we motivate users to participate actively in their communities?

Giving rewards to active users is a common approach to increasing user participation in online communities. Rewards can be in various forms, including getting better services, more privileges, or a higher status or visibility in the community [48].

Because one's behaviour is usually influenced by others, another approach to motivating users is by promoting social awareness within a community [47, 49]. The basic idea is to give information about the structure and activities in a community. Examples include displaying a list of online users, giving notifications about some events, and visualizing the users' reputation and social networks. By knowing that there are activities in their community, users are expected to stay active in it (i.e., to keep using the system). Furthermore, knowing the "presence" of others can give a good feeling to users [46]. Social awareness can also serve as social control [50] because when users realize that others know about their activities or behaviour, they will likely behave according to social norms.

6.3 Communities of Practice

In practice, the most effective ways of creating and communicating knowledge are not through written documents, but through personal interactions, including informal conversation, storytelling, and dialogue [45, 51]. Personal interactions are important because knowledge is not an isolated entity: knowledge is a concrete form of what is in a person's mind, and how others perceive it [51]. Thus, knowledge exchange processes involve both knowledge creators and users. When reading a document, readers can only rely on their perception. They cannot validate whether they have understood the document correctly. Interactions with the author can reduce this problem, as readers can argue, ask questions, or validate their understanding with the author. At the same time, the author can assess readers' interest and understanding by looking at their gestures or asking for feedback.

Because of the nature of knowledge exchange processes, people use knowledge databases not only to find specific information, but also to interact with people who produce or use the information [50]. For example, to increase the chance of getting a project proposal approved, it is very helpful to read examples of

successful proposals and to talk with the authors. The authors can give many insights that are not available in the proposals, or help introduce key persons as potential referees. Thus, in addition to storing information, knowledge databases should help users maintain and expand their social networks [49].

7 Concluding Remarks

To design a collaboration infrastructure, we should consider both its technical and social aspects. Technologies are a prerequisite for implementing such infrastructures. However, whether users will adopt and use an infrastructure often depends on design subtleties, such as how well the design fits the existing user practices. While we cannot control the dynamics of communities, we can support them by designing an infrastructure that fosters characteristics of successful communities.

In this paper, we focused mainly on personal information management and proposed design criteria for Semantic Desktop systems. However, we also brought social issues in knowledge management to the discussion, as these factors are crucial in the success of communities.

Acknowledgments. We thank Joyce Boedianto, Jeff Smith, and Yao Wang for giving comments and suggestions on drafts of this paper. The development of design criteria for Semantic Desktop systems is based on a suggestion from an anonymous reviewer.

References

1. Kollock, P.: Design principles for online communities. In: Harvard conf. on the Internet and society. (1996)
2. Davies, J.: Supporting virtual communities of practice. Industrial Knowledge Management (2001)
3. Decker, S., Frank, M.: The networked semantic desktop. In: Workshop on application design, development and implementation issues in the semantic web. (2004)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
5. Huynh, D., Karger, D.R., Quan, D.: Haystack: a platform for creating, organizing and visualizing information using RDF. In: Semantic Web Workshop. (2002)
6. Sauermann, L.: The gnowsis semantic desktop for information integration. In: 1st Workshop on Intelligent office appliances. (2005)
7. Schmitz, C.: Towards self-organizing communities in peer-to-peer knowledge management. In: Workshop on Ontologies in peer-to-peer communities. (2005)
8. Malone, T.W.: How do people organize their desks? Implications for the design of office information systems. ACM Trans. Inf. Syst. **1** (1983) 99–112
9. Barreau, D., Nardi, B.A.: Finding and reminding: file organization from the desktop. SIGCHI Bull. **27** (1995) 39–43
10. Mander, R., Salomon, G., Wong, Y.Y.: A 'pile' metaphor for supporting casual organization of information. In: Proc. conf. on Hum. factors in comp. sys. (1992) 627–634

11. Ravasio, P., Schär, S.G., Krueger, H.: In pursuit of desktop evolution: user problems and practices with modern desktop systems. ACM Trans. Comput.-Hum. Interact. **11** (2004) 156–180
12. Boardman, R., Sasse, M.A.: "Stuff goes into the computer and doesn't come out": a cross-tool study of personal information management. In: Proc. conf. on Hum. factors in comp. sys. (2004) 583–590
13. Abrams, D., Baecker, R., Chignell, M.: Information archiving with bookmarks: personal web space construction and organization. In: Proc. conf. on Hum. factors in comp. sys. (1998) 41–48
14. Whittaker, S., Sidner, C.: Email overload: exploring personal information management of email. In: Proc. conf. on Hum. factors in comp. sys. (1996) 276–283
15. Kidd, A.: The marks are on the knowledge worker. In: Proc. conf. on Hum. factors in comp. sys. (1994) 186–191
16. Kwasnik, B.: How a personal document's intended use or purpose affects its classification in an office. In: Proc. 12th ACM SIGIR conf. on Research and development in information retrieval. (1989) 207–210
17. Rodden, K., Wood, K.R.: How do people manage their digital photographs? In: Proc. conf. on Hum. factors in comp. sys. (2003) 409–416
18. Lansdale, M.: The psychology of personal information management. Applied Ergonomics **19** (1988) 55–66
19. Chase, W.G., Simon, H.A.: Perception in chess. Cognitive Psychology (1973) 55–81
20. Bondarenko, O., Janssen, R.: Documents at hand: learning from paper to improve digital technologies. In: Proc. conf. on Hum. factors in comp. sys. (2005) 121–130
21. Whittaker, S., Hirschberg, J.: The character, value, and management of personal paper archives. ACM Trans. Comput.-Hum. Interact. **8** (2001) 150–170
22. Nardi, B.A., Whittaker, S., Schwarz, H.: Networkers and their activity in intensional networks. Computer Supported Cooperative Work **11** (2002) 205–242
23. Whittaker, S., Jones, Q., Nardi, B., Creech, M., Terveen, L., Isaacs, E., Hainsworth, J.: ContactMap: organizing communication in a social desktop. ACM Trans. Comput.-Hum. Interact. **11** (2004) 445–471
24. Bellotti, V., Ducheneaut, N., Howard, M., Smith, I.: Taking email to task: the design and evaluation of a task management centered email tool. In: Proc. conf. on Hum. factors in comp. sys. (2003) 345–352
25. Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. Commun. ACM **30** (1987) 964–971
26. Simone, C., Mark, G., Giubbilei, D.: Interoperability as a means of articulation work. In: Proc. conf. on Work activities coord. and collaboration. (1999) 39–48
27. Dourish, P., Lamping, J., Rodden, T.: Building bridges: customisation and mutual intelligibility in shared category management. In: Proc. intl. ACM SIGGROUP conf. on Supporting group work. (1999) 11–20
28. Berlin, L.M., Jeffries, R., O'Day, V.L., Paepcke, A., Wharton, C.: Where did you put it? Issues in the design and use of a group memory. In: Proc. conf. on Hum. factors in comp. sys. (1993) 23–30
29. Gifford, D.K., Jouvelot, P., Sheldon, M.A., O'Toole, J.W.: Semantic file systems. In: Proc. 13th ACM symp. on Operating systems principles. (1991) 16–25
30. Gopal, B., Manber, U.: Integrating content-based access mechanisms with hierarchical file systems. In: Proc. 3rd symp. on Operating systems design and implementation. (1999) 265–278
31. Dourish, P., Edwards, W.K., LaMarca, A., Salisbury, M.: Presto: an experimental architecture for fluid interactive document spaces. ACM Trans. Comput.-Hum. Interact. **6** (1999) 133–161

32. Bowman, C.M., Dharap, C., Baruah, M., Camargo, B., Potti, S.: A file system for information management. In: Proc. ISMM intl. conf. on Intelligent information management systems. (1994)
33. Adar, E., Kargar, D., Stein, L.A.: Haystack: per-user information environments. In: Proc. 8th intl. conf. on Information and knowledge management. (1999) 413–422
34. Dourish, P., Edwards, W.K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D.B., Thornton, J.: Extending document management systems with user-specific active properties. ACM Trans. Inf. Syst. **18** (2000) 140–170
35. Hopkins, I., Vassileva, J.: Beyond keywords and hierarchies. Journal of Digital Information Management **3** (2005) 139–145
36. Karger, D.R., Quan, D.: Collections: flexible, essential tools for information management. In: CHI '04 extended abstracts. (2004) 1159–1162
37. Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., Robbins, D.C.: Stuff I've seen: a system for personal information retrieval and re-use. In: Proc. 26th ACM SIGIR conf. on Research and development in information retrieval. (2003) 72–79
38. Freeman, E., Gelernter, D.: Lifestreams: a storage model for personal data. SIGMOD Rec. **25** (1996) 80–86
39. Rekimoto, J.: Time-machine computing: a time-centric approach for the information environment. In: Proc. 12th annual ACM symp. on User interface software and technology. (1999) 45–54
40. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum.-Comput. Stud. **43** (1995) 907–928
41. Soules, C.A.N., Ganger, G.R.: Why can't I find my files? New methods for automating attribute assignment. In: Proc. 9th Workshop on Hot topics in operating systems. (2003)
42. Gemmell, J., Bell, G., Lueder, R., Drucker, S., Wong, C.: MyLifeBits: fulfilling the memex vision. In: Proc. 10th ACM intl. conf. on Multimedia, New York, NY, USA, ACM Press (2002) 235–238
43. Fisher, D., Dourish, P.: Social and temporal structures in everyday collaboration. In: Proc. conf. on Hum. factors in comp. sys. (2004) 551–558
44. Garton, L., Haythornthwaite, C., Wellman, B.: Studying online social networks. Doing Internet Research (1999) 75–105
45. Thomas, J.C., Kellogg, W.A., Erickson, T.: The knowledge management puzzle: human and social factors in knowledge management. IBM Systems Journal **40** (2001) 863–884
46. Nardi, B.A., Whittaker, S., Bradner, E.: Interaction and outeraction: instant messaging in action. In: Proc. ACM conf. on CSCW. (2000) 79–88
47. Ackerman, M.S., Starr, B.: Social activity indicators: interface components for cscw systems. In: Proc. 8th annual ACM symp. on User interface and software technology. (1995) 159–168
48. Cheng, R., Vassileva, J.: User motivation and persuasion strategy for peer-to-peer communities. In: Proc. 38th Hawaii intl. conf. on System sciences. (2005)
49. Bush, A.A., Tiwana, A.: Designing sticky knowledge networks. Commun. ACM **48** (2005) 66–71
50. Erickson, T., Kellogg, W.A.: Social translucence: an approach to designing systems that support social processes. ACM Trans. Comput.-Hum. Interact. **7** (2000) 59–83
51. Walsham, G.: Knowledge management: The benefits and limitations of computer systems. European Management Journal **19** (2001) 599–608

IRIS: Integrate. Relate. Infer. Share.

Adam Cheyer, Jack Park, Richard Giuli

SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
<FirstName>.<LastName>@sri.com

Abstract. In this paper we introduce a new semantic desktop system called IRIS, an application framework for enabling users to create a "personal map" across their office-related information objects. Built as part of the CALO Cognitive Assistant project, IRIS represents a step in our quest to construct the kinds of tools that will significantly augment the user's ability to perform knowledge work. This paper explains our design decisions, progress, and shortcomings. The IRIS project has grown from the past work of others and offers opportunities to augment and otherwise collaborate with other current and future semantic desktop projects. This paper marks our entry into the ongoing conversation about semantic desktops, intelligent knowledge management, and systems for augmenting the performance of human teams.

1 Introduction

Charles Bourne and Douglas Engelbart open their 1958 paper, "Facets of the Technical Information Problem," [1] with:

> *RECENT world events have catapulted the problem of the presently unmanageable mass of technical information from one that should be solved to one that must be solved. The question is receiving serious and thoughtful consideration in many places in government, industry, and in the scientific and technical community.*

If networked computers will be the "printing presses of the twenty-first century" and beyond, then networked *semantic desktop* applications will be the workstations of many of those knowledge workers mentioned by Stefan Decker and Martin Frank in their 2004 paper, "The Social Semantic Desktop" [6]. Today, knowledge workers are accustomed to the use of applications such as email, calendar, word processing, spreadsheets, and more. Each of those applications can be viewed as stand-alone entities, each facilitating the accomplishment of some particular task, but in no particular sense *integrated* in ways we shall call *semantic* with each other. An appropriate interpretation of John Stuart Mill's 1873 [14] call for fundamental changes in our modes of thought would suggest that we look at *semantic integration* of the tools with which we perform knowledge work.

In this paper, we will introduce a new semantic desktop system called IRIS (for Integrate-Relate-Infer-Share) and explain the context in which it was built, as part of the CALO Cognitive Assistant project. As we describe our quest to construct the kinds of tools that will significantly enhance the desktop user's experience and augment the user's ability to perform knowledge work, we will explain our design decisions, progress, and shortcomings.

2 Background and Requirements

> *I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.*
>
> –Alan Turing, *Computing Machinery and Intelligence* 1950

Before discussing the IRIS project, we will briefly describe CALO[1], an artificial intelligence application for which IRIS serves as the semantic desktop user interface. Requirements from the CALO program have greatly influenced our design for IRIS.

IRIS has been developed as part of SRI's CALO[2] project, one of two projects funded under DARPA's "Perceptive Assistant that Learns" (PAL) program[3]. The goal of the PAL program is to develop an enduring personal assistant that "learns in the wild," evolving its abilities more and more through automated machine learning techniques rather than through code changes. DARPA expects the program to generate innovative ideas that result in new science, new and fundamental approaches to current problems, and new algorithms and tools, and to yield new technology of significant value to the military and commercial sectors. Led by SRI International, 250 researchers and developers from 25 universities and companies are working on CALO.

CALO is a cognitive software system that can reason, learn from experience, be told what to do, explain what it is doing, reflect on its experience, and respond robustly to surprise. CALO's mission is to serve its user as a personal assistant, collaborating in all aspects of work life: organizing information; preparing information artifacts; mediating person-person interactions; organizing and scheduling in time; monitoring and managing tasks; and acquiring and allocating resources.

To understand and reason about the dynamics of the user's work life, CALO requires a semantically coherent view into the user's life and a mechanism for interacting with the user in a natural work setting. Our solution was to outfit CALO researchers with a semantic desktop, called IRIS, that enables them to outline the key ele-

[1] CALO: http://www.ai.sri.com/software/CALO

[2] CALO is an acronym for "Cognitive Assistant that Learns and Organizes." CALO's name was also inspired by the Latin word *calonis*, which means "soldier's servant" and conjures an image of Radar O'Reilly from the M*A*S*H TV series.

[3] DARPA's PAL program: http://www.darpa.mil/ipto/programs/pal/

ments in their environment, specifically: the projects the user works on; the key participants in various roles for these projects; the way in which accessed information relate to people, projects, and tasks in the user's life; and the priorities of tasks, messages, documents, and meetings. CALO plays the role of a collaborative teammate participating in this exercise, learning how to populate much of the semantic content and relationships on behalf of the user and the rest of his team.

In approaching the design and development of IRIS, we took much inspiration from the work of Douglas Engelbart. While Ted Nelson's Xanadu[4] [15] was arguably the first project to set the stage for modern hyperdocument processors, Engelbart's Augment[5] was the first system to actually find engagement in group document processing and sharing. In 1968, at the Fall Joint Computer Conference in San Francisco, Engelbart demonstrated Augment before a live audience.[6] Augment displayed many of the capabilities we now want to build into modern semantic desktop applications. Augment, the program, saw commercial application, and is still used today by Dr. Engelbart in his day-to-day activities. Efforts are under way to create open source variants of the Augment system [19]. At the same time, work continues on the development of an Open Hyperdocument System [9] guided by Dr. Engelbart.

CALO, Engelbart's work, the paper by Decker and Frank [6], and an earlier paper by Gradman [10] combine to provoke background thoughts that drive the evolution of our requirements. Here are several that animate the IRIS project:

1. *"Real" enough to do daily work.* As dictated by Engelbart's notion of bootstrapping,[7] we should develop using that which we are developing. To convince people to give up their current mail program, web browser, or calendar in favor of IRIS, we will need to offer a full-featured experience that supports all of their specific needs: mail encryption, spam filters, calendar servers, synchronization with PIMs[8], embedded flash, etc. Rather than implement these features ourselves, we opt to find and integrate the most mature third-party applications available into our developments.
2. *Implemented in, or able to easily integrate with, Java*. This requirement comes from the fact that many of the machine learning components we will include from CALO researchers are implemented in Java.
3. *Ontology-based knowledge store*. We require the ability to model rich semantic structures that can capture every aspect of a user's work environment.
4. *Capable of supporting organization of personal knowledge assets*. This implies providing for the ability of users to organize their information resources

[4] Xanadu: http://xanadu.com/
[5] NLS/Augment at the Computer History Museum: http://community.computerhistory.org/scc/projects/nlsproject/
[6] Videos of the first online document editing project. Found on the web at http://sloan.stanford.edu/MouseSite/1968Demo.html
[7] Engelbart's work on bootstrapping productivity: http://www.bootstrap.org
[8] Personal information managers

in ways that suit individual needs ("just for me" [4]) while maintaining semantic interoperability with other CALO installations.

5. *Cross-platform*. IRIS should be able to run on Windows, Macintosh, and Linux platforms, to support as widely as possible the diverse CALO community.

3 Related Work

With these requirements in mind, we set about looking at candidate solutions that could meet our needs. We started by looking at Mitch Kapor's Chandler[9] project, which certainly belongs in the semantic desktop category. The Chandler web site says this:

> *With Chandler, users will be able to organize diverse kinds of information for their own convenience -- not the computer's convenience. Chandler will have a rich ability not only to associate and interconnect items, but also to gather and collect related items in a single place creating a context sensitive "view" of many types of data, mixing-and-matching email, mailing lists, instant messages, appointments, contacts, tasks, free-form notes, blogs, web pages, documents, spreadsheets, slide shows, bookmarks, photos, MP3's, and so on.*

While Chandler's vision resonated with what we wanted for IRIS, its early stage of development and long product roadmap made Chandler an unsuitable starting point.

We next explored Haystack[10] from MIT. When we discovered this project [12], we were amazed how well it fit our initial designs for IRIS, in terms of both architecture and user interface design, with the added benefit of being Java-based and open source. We invited Dennis Quan to visit SRI to discuss the internals of Haystack in relation to our perceived needs. We learned much from the visit and did, indeed, begin the task of adapting Haystack's significant code base to our framework. Haystack's approach to ontology-driven architectures was to create a language, Adenine [3]. With Adenine, all user interface objects, the overall system architecture, and information assets are defined in an ontology. IRIS took a slightly different approach; an OWL ontology defines information assets. Instead of an API based on a language like Adenine, IRIS implements specialized APIs for each OWL class. This provides programmers with a convenient, object oriented access to the knowledgebase. The user interface layout for Haystack and IRIS are greatly similar; both rely on a three-column view structure, where the three concerns of navigation, focus of attention, and context are each presented in their own view. For a variety of reasons, we ended up moving in a different direction, but Haystack and Dr. Quan's deep knowledge of the subject gave us a solid start.

[9] Chandler: http://www.osafoundation.org/
[10] Haystack: http://haystack.lcs.mit.edu/

The next system we evaluated was the Radar Networks[11] Personal Radar, a very impressive semantic desktop that turned out to share many of the goals and requirements for IRIS: Java-based, ontology-driven, user centric. We negotiated, and CEO Nova Spivack agreed to join the CALO project to help combine elements of Personal Radar into the IRIS code-base. In particular, we adopted their Semantic Object framework, a very fast triple-store implementation, and certain elements from their SWING-based user interface. These are described in more detail in Section 4.

Well down the path of implementing IRIS, we came across Gnowsis[12]. Gnowsis [18] appears to offer integration with many of the same third-party applications as IRIS, and to share many similar philosophies regarding application and data integration. Where IRIS and Gnowsis currently diverge may lie in the way in which those applications are integrated. Whereas Gnowsis appears to have fairly loose integration with standalone applications, using adapters to copy references from applications into a local server and a separate browser for navigating and searching the data, IRIS has chosen to be more tightly integrated at the user interface level, providing an "embedded suite" of applications. Each plug-in application is instrumented such that IRIS captures semantic events as they occur. For example, IRIS "knows" which web page is being browsed, or which email has been opened for reading. Tight integration of applications is particularly useful to IRIS's learning framework and components, which can offer real-time suggestions as the user works with information. Having said that, it is worth noting that, as work on IRIS progresses, we are beginning to relax the tight integration between the presentation layer and the backside. Indeed, notions of external presentation elements are now under consideration.

Most recently, we discovered MindRaider[13], a project arguably close to IRIS, Haystack, and Gnowsis in spirit and intent. MindRaider's open source license precludes us from looking closely at the source code, but we observe, while running the program, that there are profound similarities between its ontology-driven architecture and that of IRIS. Without examining implementation, we suspect that a central ontology is at work determining classes of information assets and constraining relationships between those classes. We also note the same interesting parallels in user interface design as mentioned with Haystack. We suspect that HCI (user interface design) will eventually rise to be at least as important to the success of semantic desktops as is semantic interoperability among platforms.

Topic Maps [17] is another research area we are tracking, as we consider IRIS a kind of topic map for personal information assets. A topic map[14] provides a container for proxies for subjects, called topics. Each subject, which is anything that can be the focus of thought or discussion, is represented by one topic. Each topic is a kind of container for links to all known information about the subject. Topics have properties that provide subject identity and other properties of the subject. Topics can play roles

[11] RadarNetworks: http://www.radarnetworks.com/
[12] Gnowsis: http://www.gnowsis.org/
[13] MindRaider: http://mindraider.sourceforge.net/
[14] Topic maps: http://www.topicmaps.org/

in association with other topics. For instance, a topic, which is a proxy for the IRIS user, can play roles such as member in a meeting, speaker at a conference, parent or spouse, and more. Topics are associated with occurrences. For instance, the topic for a particular personal computer can be linked with occurrences such as web pages where that computer can be purchased, or where an online help system is found. Topics are also repositories for all possible ways to *name* the topic. With this, individuals can assign names for things in their personal space; that personal space thus gains the "just for me" [4] flavor. Since IRIS is an "ontology-driven" platform, the addition of a topic map structure to the IRIS knowledgebase facilitates this "just for me" characteristic. User assigned names and relationships can be added without affecting the IRIS ontology.

4 The IRIS Semantic Desktop

IRIS is an application framework for enabling users to create a "personal map" across their office-related information objects. IRIS is an acronym for "Integrate. Relate. Infer. Share." In the following text, we will adopt these four terms as organizing subsections, as we describe IRIS's design, architecture, implementation, and use-cases.

4.1 IRIS – Integrate

> *Hypertext is a form of storage, a new form of literature, and a network that just might revitalize human life.*
>
> –Ted Nelson 1965

IRIS is first and foremost an integration framework. Whereas in today's packaged applications suites, where only loose data integration exists[15] (usually limited to the clipboard and common look-and-feel for menus and dialog boxes), IRIS strives to integrate data from disparate applications using reified semantic classes and typed relations. For instance, it should be possible to express that "*File F* was presented at *Meeting M* by *Tom Jones*, who is the *Project Manager* of *Project X*," even if the file manager, calendar program, contact database, and project management software are separately-developed third-party applications. In a Topic Maps fashion, there should be a single instance that represents each concept, and all that is knowable about that concept should be directly accessible from that instance [17].

The IRIS framework offers integration services at three levels (Figure 1):

1. Information resources (e.g., an email message, a calendar appointment) and the applications that create and manipulate them must be made accessible to

[15] Even within a single application, deep data integration is usually pretty threadbare. Consider Microsoft Outlook: the email addresses displayed in a message are not linkable (or deeply related) to the people records in your contacts folder.

IRIS for instrumentation, automation, and query. IRIS offers a plug-in framework in the style of the Eclipse[16] architecture or the JPF framework[17], where "applications" (components with a graphical user interface) and "services" (processing components with no GUI of their own) can be defined and integrated within IRIS. Apart from a very small, lightweight kernel, all functionality within IRIS is defined using the plug-in framework, including user interface, applications, back-end persistence store, learning modules, harvesters, and so forth.

2. A knowledge base (KB) provides the unified data model, persistence store, and query mechanisms across the information resources and semantic relations among them. Ontology services are described in more detail in Section 4.2, Relate.

3. The IRIS user interface framework allows plug-in applications to embed their own interfaces within IRIS, and to interoperate with global UI services such as the notification pane, menu and toolbar management, query interfaces, the link manager, and suggestion pane.

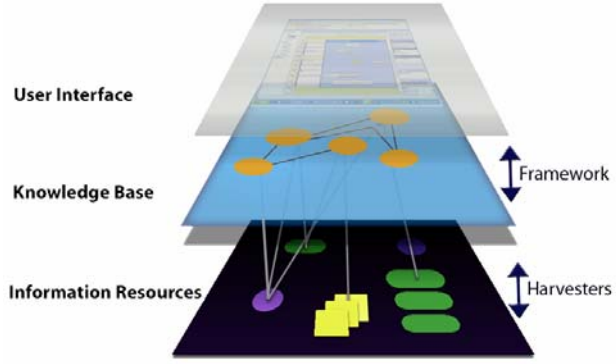


Figure 1: The three-layer IRIS integration framework.

IRIS comes "out of the box" with several integrated office applications:

- *Email*: After initially integrating Java-based Columba,[18] we moved to Mozilla[19] for email, as it is one of the most popular, full-featured, cross-platform applications available. To integrate Mozilla with Java, we adopted and made significant extensions to the JREX[20] package, and then ran Email as an embedded XUL[21] application.

[16] Eclipse: http://www.eclipse.org/
[17] Java Plugin Framework (JPF) Project: http://jpf.sourceforge.net/
[18] Columba Mail: http://columba.sourceforge.net/
[19] Mozilla Application Suite: http://www.mozilla.org
[20] JREX – Mozilla through Java: http://jrex.mozdev.org/
[21] XUL – http://www.xulplanet.com/

- *Web browser*: Mozilla provides an much better web browsing experience than our initial integration effort, Java-based CALPA.[22]
- *Calendar*: We selected OpenOffice GLOW[23] because it is Java-based and iCAL compliant, interoperates with the Sun/Netscape calendar server used by SRI, and has a very nice user interface. We believe there remains room for improvement in our calendar application.
- *Chat*: We implemented our own interface to the Jabber[24] instant messaging backend.
- *File explorer*: We wrote our own in Java.
- *Data editor/viewer*: To view and edit data records such as people, projects, tasks, and any other ontology-based object in the KB, we used a forms package from Radar Networks' Personal Radar software.

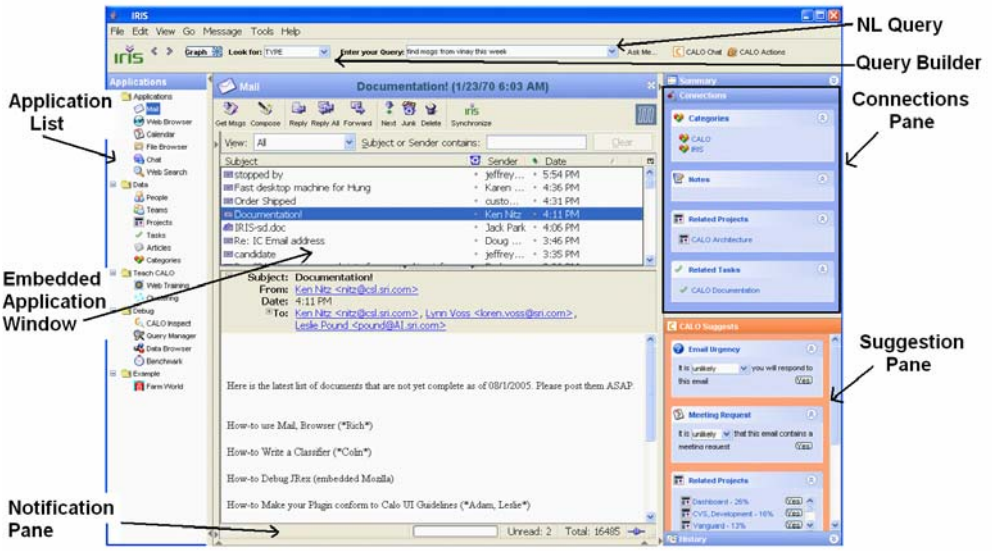


Figure 2: The IRIS user interface.

The IRIS user interface provides the "shell" for hosting each of these embedded applications (figure 2). Two collapsible side panels frame the main application window, one for selecting among available applications, the other for displaying and editing semantic links for the selected application object and presenting contextual suggestions from the learning framework. Applications can add toolbars to the IRIS frame, and when selected, an application's menu items are "merged" with IRIS menu functionality present for all applications. IRIS provides an extensible context-

[22] CALPA: http://htmlbrowser.sourceforge.net/
[23] OpenOffice GLOW calendar: http://groupware.openoffice.org/glow/
[24] Jabber Instant Messaging: http://www.jabber.org/

sensitive online help system and several methods for querying information resources within and across applications. An example of a natural language query supported by iAnywhere's Answers Anywhere[25] IRIS plugin is "find email from Vinay last week related to the CALO project."

4.2 IRIS – Relate

> *Information is both more and less real than the material universe. It's more real because it will survive any physical change; it will outlast any physical manifestation of itself. It's less real because it's ineffable. For example, you can touch a shoe, but you can't touch the notion of "shoe-ness" (that is, what it means to be a shoe). The notion of shoe-ness is probably eternal, but every shoe is ephemeral.*
>
> – Steven R. Newcomb [17, page 32]

IRIS is used to *semantically integrate* the tools of knowledge work. What do we mean by this? We use the term "semantic" in the sense used by the Semantic Web community, where markup technologies are being wedded to the tools of semantic representation (e.g., ontologies, OWL, RDF). This facilitates putting data on the web in such a way that machines can access it, make meaningful references to it, and perform manipulations on it, including reasoning and inference. In that sense, IRIS provides a knowledge representation by which the artifacts of a user's experience such as email messages, calendar events, files on the disk or found on the web, can be stored and related to each other across applications and across users.

When defining the ontology to be used for IRIS, a design choice had to be made: Do we use a small, simple ontology or a complex, more-expressive ontology? We first implemented a fairly large, yet straightforward, ontology. However, the requirement that IRIS interoperate with CALO's reasoning and learning capabilities drove us to adopt CALO's pre-existing ontology, which supports roles, events, and complex data structures.

CALO's ontology is called CLib,[26] the Component Library Specification, which consists of definitions for everyday objects and events, as well as axioms to support the beginnings of common-sense reasoning. For IRIS, we translate CLib, implemented in a knowledge language called KM, to OWL. We chose OWL[27] as the data representation in IRIS because it is a W3C-approved standard that allows for a flexible data schema and query that supports inheritance. Currently IRIS supports the OWL Lite subset, with future plans to support OWL DL.

---

[25] Answers Anywhere NL query: "http://www.ianywhere.com/products/answers_anywhere.html

[26] KM Component Library: http://www.cs.utexas.edu/users/mfkb/RKF/tree/ then select "Core+Office" to browse the CALO subset.

[27] Web Ontology Language: http://www.w3.org/TR/owl-features/

IRIS provides a framework for *harvesting* application data and *instrumenting* user actions in IRIS applications. The harvesting of data refers to importing external data into semantic (ontology-based) structures. For example, if given the specification of an email instance, harvesting APIs exist to create ontology structures for the email, addresses, and people associated with those addresses. These ontology structures are available in the instrumentation API for application events. This data is then translated once again to an external event publish/subscribe model that allows other IRIS plug-ins or external applications to access the data.

4.3 IRIS – Infer

> *If you invent a breakthrough in artificial intelligence, so machines can learn, that is worth 10 Microsofts.*
>
> –Bill Gates, quoted by NY Times[28]

One of the key differentiators of IRIS, compared to many semantic desktop systems, is the emphasis on machine learning and the implementation of a plug-and-play learning framework. We see machine learning as one of the solutions around a key issue limiting the semantic web's growth and mass adoption: Who is going to enter all of the required links and knowledge?

Here we present a typical use case of how learning components integrated within the IRIS framework combine to progressively construct a semantic representation of the user's work life.

Step 1: Email Harvesting: As the user receives email in Mozilla, IRIS automatically harvests messages, adding them as semantic instances in the knowledge base. As part of this process, names in the address fields are normalized (e.g., "Rich Giuli" will match "Richard Giuli"), links are created to existing contact records in the KB, and new records are added for people not in the KB. Events indicating new email messages and people records are published to the Instrumentation Bus for other learning components to consume.

Step 2: Contact/Expertise Discovery: When contact records containing a name and email address are added to the KB, the DEX service (from UMass), a CALO component, wakes up and tries to discover additional information for that person. Contact information is discovered, as well as a "gist" representing a person's expertise, composed of keywords and noun phrases that are significant for the person.

Step 3: Learn from Files: In a similar fashion to email, IRIS harvests information from files on the user's desk. Currently, SEMEX [7] (from UWashington)

[28] Gates speech: http://www.nytimes.com/2004/03/01/technology/01bill.html

opens LaTeX, BiB, and Microsoft Office files (Word, Excel, PowerPoint) to add content (e.g., publication references) to people in the contact KB.

Step 4: Project Creation: Clustering algorithms in IRIS are applied to the user's email to propose new projects to be added to the KB. For each project instance, a label for the project is proposed using the most salient phrase in the email cluster, keywords are added that provide a "gist" of the project, and links are added to project participants using the people in the from/to fields for the email cluster. IRIS provides a user interface where the differences between multiple clustering algorithms can be explored. Currently, three algorithms have been integrated into the framework: Carrot2/Lingo, based on singular value decomposition [16]; an algorithm based on agglomerative clustering and social network analysis [11]; and an algorithm based on linear optimization with user-specified centroids.

Step 5: Classification According to Project: Leveraging the textual content and relations extracted for projects, people, and files, a Bayesian classifier is applied to hypothesize relationships between projects and objects such as emails, files, web pages, and calendar appointments. IRIS's suggestions are displayed to the user, who can optionally provide feedback to the algorithm by indicating the correct values (Figure 3).

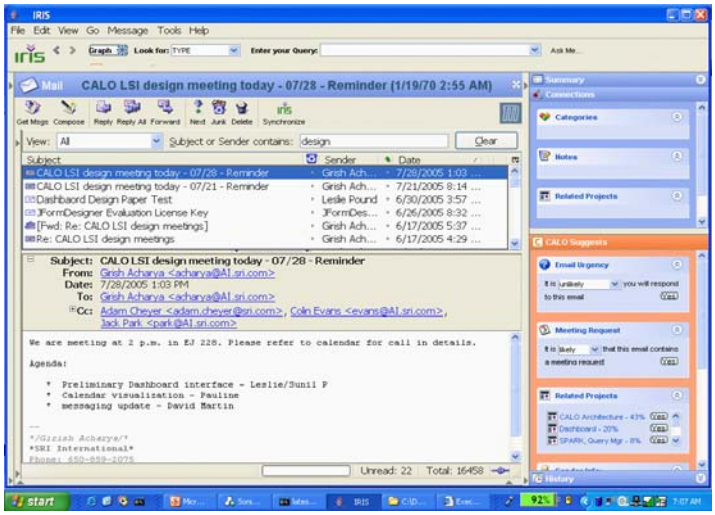


Figure 3: In the "CALO Suggests" pane, learned hypotheses about an email are presented, including reply urgency, meeting detection, project association, and others. The user can provide feedback about the system's choices, and the system will adapt accordingly.

Step 6: Higher-level Reasoning: A number of specialized reasoners within CALO continually examine events in the user's activity stream and attempt to make useful inferences. For instance, when the user clicks on an email, IRIS

attempts to predict whether the user will/should reply to the message (Figure 3). Another plug-in applies text summarization techniques to produce a gist that will be faster for the user to read. In several use cases, multiple reasoners are combined to produce a single prediction. If the user provides negative feedback to a resulting hypothesis, each individual reasoner will adapt itself accordingly, and a meta-learner will use the intermediate results from each predictor to improve its own logic about how their results are combined. Implemented examples of this approach include the "Meeting PrepPack" reasoner and the "Meeting Request" detector.

This use case gives readers a flavor of the types of learning components that have been integrated within IRIS to help construct and then leverage a semantic model representing the user's work life. We feel that we have just scratched the surface of the types of useful learning-based functionality that can be integrated into the IRIS semantic desktop, and we are eagerly anticipating continued development, working with members of the CALO and open source communities.

4.4 IRIS – Share

> *Prior to the Internet, the last technology that had any real effect on the way people sat down and talked together was the table.*
>
> –Clay Shirky, at Emerging Technology Conference 2003[29]

Sharing information is one of the four key concepts that make up the IRIS vision. We feel that the ability to learn and leverage semantic structure in organizing one's work life will be greatly enhanced in a collaborative setting. Shared structures are essential for both end-user applications, such as team decision making and project management, and for infrastructural components such as machine learning algorithms, which improve when given larger data sets to work on.

In the first version of IRIS, we experimented with a simple collaborative functionality using a Jabber-based transport mechanism. Changes to shared data were written to a "chat room" space representing an ACL group; each IRIS client would remember what changes it had seen previously, and upon startup, initialize its KB by applying all recently recorded change actions. This approach had the benefit that it supported real-time collaborative work between online participants as well as enabling "latecomers" to join and be initialized to the common state. However, with no locking mechanisms, users were plagued by inconsistencies, and we temporarily removed the collaboration feature.

In the coming year, the CALO project has planned functionality and applications that will require infrastructure to support collaborative team decision making, as well as reasoning over a shared document space. We are therefore currently revisiting what approaches to take regarding a collaboration infrastructure for IRIS.

[29] Clay Shirky: http://shirky.com/writings/group_enemy.html

5 Evaluations & Conclusions

IRIS is now in daily operation as the primary office environment used by the authors and several other members of the CALO community. In addition, as part of the formal evaluations of the CALO project, IRIS and its learning components were used extensively by 15 users during a few weeks of testing, giving CALO an opportunity to learn "in the wild" through observation and interaction with its user. After this experimentation period, we interviewed the users to understand what they liked, what features were missing, and how IRIS generally should be improved.

We were encouraged that most of the feedback was quite positive, with many of the users stating that they were generally pleased with the robustness of the system and impressed with the capabilities of IRIS to learn and provide useful data to them. In particular, the capability to automatically discover contact and "gist" information for people from whom they receive email was much appreciated. Several reports of events where IRIS made a significant positive contribution were noted. For instance, one user, after skimming a long email from his boss, wondered why IRIS was flagging the message as a meeting request. Upon closer reading, he discovered that towards the end of the message, his boss was actually requesting a meeting with him later that afternoon, with an expected deliverable.

Despite the positive feedback, a number of issues still need to be addressed before most users will be willing to adopt IRIS as their primary work environment:

1. Performance was the Number 1 issue. Many noted that the startup time for IRIS was quite slow, and actual use was sluggish, particularly during email use, where many research-quality components would process each selected message. Subsequent analysis revealed that 68% of the startup time and 72% of memory use could be attributed to three learning components; these will be candidates for optimization in the future.

2. Many user interface issues were noted, in particular regarding real-estate management for smaller screens, several inconsistencies in UI design, and the desire to use drag-and-drop. Also, several minor user interface bugs were mentioned, the most annoying being a proclivity for IRIS to pop up or become the selected window in an unsolicited way when activity (such as the arrival of an important email message) occurs. Improvements to the user interface experience will become a significant priority for the near-term roadmap.

3. Finally, many users, excited by the glimmer of intelligence that IRIS (and the cognitive assistant CALO) at times seemed to exhibit, made numerous suggestions about use cases they thought the pair would be able to help them with in the future to aid their productivity.

In the coming months, we will aggressively pursue these and other roadmap items, with a particular emphasis on making IRIS usable and useful for collaborative teams.

We remain enthusiastic about the potential for coupling semantic representations, machine learning, user interface design, and real-world office systems.

6 Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the Department of Interior-National Business Center (DOI-NBC).

We would like to thank Nova Spivack and Jim Wissner at Radar Networks for their tremendous contributions to the code base, ontologies, and vision for IRIS. Many members of the CALO LSI team worked hard to make IRIS what it is: Colin Evans, Steve Hardt, Jim Carpenter, Ken Nitz, Ayse Onalan, Leslie Pound, Girish Acharya, Mark Gondek, Talia Shaham, Julie Wong, Ken Doran, David Dunkley, Chris Brigham, and Jason Rickwald. Thanks to the CALO management team for supporting the concept: Bill Mark, Ray Perrault, David Israel, Jim Arnold, and Jeffrey Davitz. Final thanks to those non-SRI CALO members, too many to name, who are working with us to integrate their cutting-edge technologies into the IRIS learning framework.

References

1. Bourne, Charles P., and Douglas C. Engelbart, “Facets of the Technical Information Problem,” *SRI Journal*, Vol. 2, No. 1, 1958. On the web at http://bootstrap.org/augdocs/friedewald030402/facets1958/Facets1958.html
2. Bush, Vannevar, “As We May Think,” The Atlantic Monthly, July, 1945. On the web at http://www.theatlantic.com/doc/194507/bush
3. Quan, Dennis, “Metadata Programming in Adenine”. February 2003. On the web at http://haystack.csail.mit.edu/documentation/adenine.pdf
4. Park, Jack and Adam Cheyer. “Just For Me: Topic Maps and Ontologies”, submitted TMRA ’05 Topic Maps Research and Applications Workshop, Leipzig, Germany, October 6, 2005.
5. Culotta, Aron; Bekkerman, Ron; McCallum, Andrew. “Extracting Social Networks and Contact Information from Email and the Web.” In Proceedings of CEAS, First Conference on Email and Anti-Spam (CEAS). July 2004. On the web at http://www.cs.umass.edu/~culotta/pubs/ceas04.pdf
6. Decker, Stefan; and Martin Frank, “The Social Semantic Desktop,” 2004. On the web at http://www.deri.at/publications/techpapers/documents/DERI-TR-2004-05-02.pdf
7. Dong, Xin; Halevy, Alon; Nemes, Ema; Sigurdsson, Stephan. “SEMEX: Toward On-the-fly Personal Information Integration.” Workshop on Information Integration on the Web (IIWEB). Toronto, CA. August 2004. On the web at http://data.cs.washington.edu/papers/semex_iiweb.pdf

8. Engelbart, Douglas, “Augmenting Human Intellect,” October, 1962. On the web at http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html
9. Engelbart, Douglas, “Draft OHS-Project Plan,” October 23, 2000. On the web at http://www.bootstrap.org/augdocs/bi-2120.html
10. Gradman, Eric, “Distributed Social Software,” December 2003. On the web at http://www.gradman.com/projects/dss/final/final.pdf
11. Huang, Yifen; Govindaraju, Dinesh; Mitchell, Tom; Rocha de Carvalho, Vitor; Cohen, William.. “Inferring Ongoing Activities of Workstation Users by Clustering Email.” In Proceedings of CEAS, First Conference on Email and Anti-Spam (CEAS). July 2004. On the web at http://www.ceas.cc/papers-2004/149.pdf
12. Karger, David R., Karun Bakshi, David Huynh, Dennis Quan, and Vineet Sinha, “Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data,” in CIDR 2005, Asilomar, California. On the web at http://www-db.cs.wisc.edu/cidr/cidr2005/papers/P02.pdf
13. McCluhan, Marshal, *The Medium is the Message*, Wired Books, 1996.
14. Mill, John Stuart, *Autobiography*, 1873. On the web at http://www.utilitarianism.com/millauto/seven.html
15. Nelson, Theodor von Holm. “Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use.” ACM Computing Surveys 31(4), December 1999. On the web at http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/60.html
16. Osinski, Stanislaw; Stefanowski, Jerzy; Weiss, Dawid. “Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition.” Intelligent Information Systems 2004: 359-368. On the web at http://www.cs.put.poznan.pl/dweiss/site/publications/download/iipwm-osinski-weiss-stefanowski-2004-lingo.pdf
17. Park, Jack, Editor, and Sam Hunting, Technical Editor, *XML Topic Maps: Creating and Using Topic Maps for the Web*, Boston, MA. Addison-Wesley, 2003
18. Sauermann, Leo, “The Gnowsis Semantic Desktop for Information Integration,” in IOA 2005, Kaiserlauten, Germany. On the web at http://www.dfki.uni-kl.de/~sauermann/papers/Sauermann2005a.pdf
19. Thomas, Dave, “Open Augment – Back To The Future Preserving the Augment Legacy With XML,” XML 2003 Conference. Philadelphia, PA. December 2003. On the web at http://www.idealliance.org/papers/dx_xml03/papers/05-00-00/05-00-00.pdf

Harvesting Desktop Data for Semantic Blogging

Knud Möller and Stefan Decker

DERI, National University of Ireland, Galway
{knud.moeller, stefan.decker}@deri.org

Abstract. A typical computer user's desktop contains large amount of formal data, such as addresses, events or bibliopraphies. Especially within a corporate or organizational environment, it is often important to exchange this data between employees. However, state-of-the-art communication technologies such as email or bulletin boards don't allow to easily integrate desktop data in the communication process, with the effect that the data remains locked within a user's computer. In this paper, we propose that the recent phenomenon of blogging, combined with a tool to easily generate Semantic Web (SW) data from existing formal desktop data, can result in a form of semantic blogging which would help to overcome the aforementioned problem. We discuss a number of preconditions which must be met in order to allow semantic blogging and encourage users to author a semantic blog, and we present a prototype of the *semiBlog* editor, which was created with the purpose of user-friendly semantic blogging in mind. We argue that such a semantic blog editor should integrate tightly with a user's desktop environment, as this would make integration of existing data into the blog as easy as possible.

1 Introduction

Computer users generate and collect a large amount of data on their desktop (we use the term *desktop* as a metaphor for the entire working environment within a computer). While the major part of this data usually consists of unstructured resources such as texts and images, a substantial portion of it is also structured or semi-structured data — users store their contacts in electronic address books, organize their appointments in a calendar application or manage bibliographic references in a bibliography tool. Within organizations and companies, this data is often part of the organizational knowledge, and needs to be distributed within and in between project groups and employees. The process of distribution is often complicated and cumbersome, as the data tends to be locked within each individual user's desktop. Communication channels such as email or bulletin boards exist, but they do not normally offer ways to integrate existing formal data. Applications like MS Outlook[1] or MeetingMaker[2] offer data exchange for specific kinds of data, but they cannot be used as a general means for data exchange. Also, the exchange is restricted to users of these applications.

The recent phenomenon of weblogging (or "blogging") [13] has been suggested as a light-weight, general purpose tool for data exchange and knowledge management within organizations [10]. With blogging moving more and more into the mainstream of web technologies, analysts such as Forrester have now taken up the topic. A report by Forrester [6] suggests blogging as a valuable publicity tool for companies and a means to keep in touch with their customer base. Furthermore, the report stresses that blogging can also serve as an inter-company communication channel, in the sense that employees could provide each other with important data about the work

[1] Microsoft Outlook: http://www.microsoft.com/outlook/
[2] MeetingMaker: http://www.meetingmaker.com

they are doing (either through their own personal blog, or through a group blog), such as project details, etc. In a similar line of thought, [12] suggests that blogging can be used as a way to perform *Knowledge Management* within an organization or company. While these observations show an interesting use case for blogging, the data that is being exchanged in this way is currently locked within the blog (or blogosphere), and disconnected from other resources a user might have on their desktop. An employee can blog about an upcoming meeting, but there is no simple way he or she could reuse existing data from their electronic calendar, addressbook, etc. In the same way, another employee can read the entry and manually add the relevant information to the various desktop applications — e.g. when and where the meeting takes place or details about people attending it — but again there is no simple way to automatically import this data.

1.1 Contributions of this Paper

In this paper, we suggest a form of semantic blogging to overcome the limitations posed by both the ordinary blogging approach and standard applications such as Outlook or MeetingMaker, in the sense that a user can easily add metadata about the things they blog, and other users can just as easily import this metadata and in turn incorporate it into their desktop applications. The term *semantic blogging* has largely been coined by Cayzer [2]. We argue that semantic blogging as a means of data exchange can best be implemented as a desktop-based approach, as this allows better access to and tight integration with the relevant data, a thing that is difficult or impossible in web-based approaches. Desktop applications are also capable of providing greater user-friendliness, as they (ideally) behave within parameters set by the UI paradigm of the user's desktop platform and commonly allow the use of visual techniques such as drag-and-drop. As a proof-of-concept for our approach, we have built a desktop-based blog editor called "semiBlog", as well as an accompanying blog reader.

1.2 Outline of the Paper

We will begin with a short introduction to semantic blogging and blogging in general in section 2, followed by an illustration of how semantic blogging could be used as a means for the exchange of formal desktop data in section 2.1. In section 2.2 we will then discuss a number of preconditions which have to be met in order to allow this kind of semantic blogging, and to make it as user-friendly as possible. Section 3 will present our implementation of semantic blogging, the *semiBlog* application, starting with a simple use case and subsequently diving deeper into the actual design of the system. Finally, we will end the paper with a look at other approaches at semantic blogging, as well as related technologies in section 4.

2 Semantic Blogging

Blogging has made it possible for ordinary users to publish on the Web, and thus become content producers instead of content consumers. A number of web-based blogging platforms such as Blogger[3] or Movable Type[4] allow users to publish various kinds of data (though most blogs consist mainly of text and pictures). Blog authors manage their own content in their own blog, structure it through time in the form of discrete blog entries and are often able to categorise these entries. Usually, blogs

[3] Blogger: `http://www.blogger.com`
[4] Movable Type: `http://www.sixapart.com/movabletype/`

also offer so-called newsfeeds, which are essentially XML-based tables of contents. Using these newsfeeds, users can subscribe to blogs and aggregate information from different blogs. Finally, bloggers are able to comment on posts in other people's blogs or refer to them through links (trackbacks[5], pingbacks[6]), thus creating what is called the *blogosphere*.

Semantic blogging extends the blogging paradigm with the possibility to add additional metadata to a blog entry. This metadata can cover a variety of aspects, ranging from information about the entry itself, such as the author or date of publication (using vocabularies such as RDF Site Summary (RSS 1.0)[7]), information about the structure of the blog and relations to other communication sites (e.g. by using the SIOC [1] ontology), to metadata describing the topics mentioned in the entry (a blog entry about a meeting would e.g. include metadata about that meeting, such as date and location, people attending or details of project related to the meeting, whereas an entry discussing a book would include bibliographic metadata for that book). Moreover, this metadata is expressed in a semantic format such as the Resource Description Framework (RDF), which allows further inferencing and aggregation.

2.1 Semantic Blogging as Data Exchange

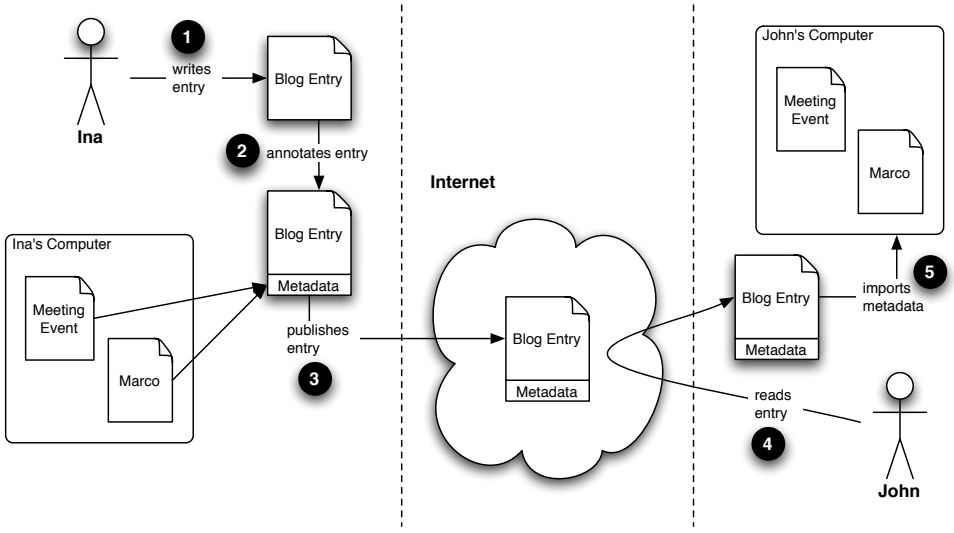


Fig. 1: Exchanging data through a blog

A specific application of semantic blogging is its use as a means of data exchange between users, e.g. employees in a company. A user would enrich a blog entry with metadata originating from some existing desktop data. Readers of the blog could then import this metadata into the appropriate applications on their desktop. Figure 1 illustrates this idea. A user named Ina wants to inform her colleagues about an upcoming project meeting. Instead of writing an email, and sending it to a specific set of people, she creates a blog entry announcing the meeting. Because she already

[5] Trackbacks: `http://www.movabletype.org/trackback/`
[6] Pingbacks: `http://www.hixie.ch/specs/pingback/pingback/`
[7] RSS 1.0: `http://web.resource.org/rss/1.0/`

has the relevant data (when and where) in her calendar application, she can reuse this data and attach it as metadata to the blog entry. Similarly, she might want to tell her colleagues that a visitor from another company (Marco) is going to join the meeting, and adds Marco's address book entry to the blog. In this way, semantic metadata can be made available with little or no overhead. Among other people interested in this project, an employee namend John has earlier subscribed to the blog. He reads the entry and decides to go to the meeting. Instead of scribbling down the relevant data, he just imports it into his calendar and address book applications, where it is now ready to be used.

2.2 Preconditions for Semantic Blogging

A semantic blogging system in general, and more specifically a system that wants to offer the kind of semantic blogging functionality outlined in section 2.1, would have to meet certain preconditions, which will be discussed in the following sections.

- **Reference Mechanism** In order to reference and access desktop objects, a suitable reference mechanism has to be devised.
- **Metadata Representation Format** A suitable format has to be chosen to represent semantic metadata within the blog.
- **Usability** Since authoring a semantic blog should be just as easy as authoring an ordinary blog, a blogging platform has to be designed to maximize ease of use and user-friendliness.

Referencing Objects on the Desktop If we want to be able to integrate data objects from a user's desktop into a blog, we have to able to somehow reference and access these objects. While it would be possible to import the complete metadata for some desktop object (e.g. someone's addressbook entry) once during the creation of a new blog entry, this would mean that the blog entry would only reflect the desktop object at that particular point in time. Subsequent changes to the desktop object (e.g. changing a person's phone number) would be invisible to the blog. Therefore, a simple referencing mechanism is needed, so that the blog reflects the current state of a desktop object every time it gets published. One possible solution is a URL-based reference mechanism, as URLs are well understood and supported. In principle, however, any other kind of reference mechanism can be imagined as well. It should be noted that references to desktop objects are only relevant internally as *locators* within the blog-authoring platform — blog consumers would not see these references, but instead see a formal representation of the thing that is described by the desktop object. I.e. the metadata attached to a blog entry is not about the desktop object from which it was generated, but about the thing that this desktop object describes.

Technically, the reference mechanism could be implemented in a variety of ways. Ideally, the desktop environment would already provide this. However, while research and development in the area of semantic desktops (which would provide each object on the desktop with a URL) such as the "Gnowsis" platform [11] has already come a long way, semantic desktops are still far from being mainstream, and can therefore not be assumed to be accessible to a user wishing to author a semantic blog. Another possibility are metadata-enabled file systems, such as the one that Mac OS X 10.4 ("Tiger")[8] offers. Here, every file object (and, by means of a work-around, even non-file objects) can have arbitrary metadata attached, which effectively means a semantic desktop "light". Finally, a solution that doesn't impose any requirements on the underlying desktop system is the definition of interfaces

[8] Tiger: `http://www.apple.com/macosx/`

for different kinds of data within the blogging platform itself. We have chosen this solution for our *semiBlog* application, but will look into support for Gnowsis-type systems in the future.

Metadata Representation Format While software like MS Outlook or MeetingMaker already allow the exchange of specific kinds of data, they use static, proprietary formats and bind a user to a specific piece of software. Within our vision of semantic blogging, however, we want to allow the user to exchange *any kind of data*, regardless of the software they use. To represent metadata in a blog, we would therefore need an open, application independent format. To accomodate any kind of data, we also need a format that allows to represent arbitrarily complex objects. Finally, the format should have a means to reference and link to other objects. Considering this list of requirements, we believe that RDF would be a good choice for a representation format. Generally, the graph structure of RDF (as opposed to the tree structure of simpler formats like XML) makes it more adept at expressing semantic relations and metadata.

```
<rdf:RDF
    <!-- ... namespace declarations,
        metadata about the blog in general, etc. ... --> >
    <rss:item rdf:about=
        "http://www.example.org/blog#YARSMeeting">
        <rss:title>YARS and Space Travel</rss:title>
        <rss:link>
            http://www.example.org/blog#YARSMeeting
        </rss:link>
        <rss:description>
            Today I had a meeting with Andreas. We went over
            his paper and talked about possibilities of using
            his YARS RDF store for a manned mission to Alpha
            Centauri. ...
        </rss:description>
        <dc:date>2005-04-06</dc:date>
        <dc:subject>
            <foaf:Person rdf:ID="andreas">
                <foaf:homepage>http://sw.bla.org/~aharth/</foaf:homepage>
                <foaf:surname>Harth</foaf:surname>
                <foaf:firstName>Andreas</foaf:firstName>
                <!-- ... more properties ... -->
                <rdf:value>Andreas Harth</rdf:value>
            </foaf:Person>
        </dc:subject>
        <dc:subject>
            <bibtex:InProceedings>
                <bibtex:title>Yet Another RDF Store: Perfect Index
                Structures for Storing Semantic Web Data With
                Contexts</bibtex:title>
                <bibtex:author rdf:resource="#andreas" />
                <!-- ... more properties ... -->
                <rdf:value>YARS Paper</rdf:value>
            </bibtex:InProceedings>
        </dc:subject>
    </rss:item>
    <!-- ... more entries ... -->
</rdf:RDF>
```

Fig. 2: Example of an RSS 1.0 feed, enriched with additional metadata

Using RDF also makes the technical side of adding metadata to the blog easier. Blogs usually provide a table of contents in the form of a newsfeed. These feeds come in a variety of formats, one of which is RSS 1.0. Due to the open nature of RDF, it is possible to add arbitrary additional RDF triples to such a feed, without breaking its validity. Figure 2 shows an excerpt from a newsfeed in RSS 1.0,

using `RDF/XML` syntax[9]. Each entry in the feed (here only one) is represented by an object of type `rss:item`. Using the `dc:subject` property, additional metadata has been added (the example shows metadata about a person using the Friend of a Friend (FOAF)[10] vocabulary, and about a publication using a vocabulary which implements the BibTEX format).

Usability Authoring a semantic blog should be made just as easy as authoring an ordinary blog. While the metadata added to a blog entry could in principle be hand coded or added through specific form fields in a web-based blog editor, we believe that this would be far too complicated to appeal to a non-technical user, like the average employee we are aiming at. As noted in the article by Hendler [4], semantic metadata should be produced as a by-product of tasks that a user is already used to perform on a day-to-day basis, such as entering people in an address book application, organizing events in a calendar or managing publications in a bibliographic database. We believe that the easiest way to allow access to this data from a blogging platform is to implement it as a desktop application. Since a (good) desktop application also integrates nicely into the UI paradigms of its specific platform, this will guarantee that the user feels at home in the application. While a platform-independent solution would certainly be desirable, we believe that it will always be necessary to implement parts or the whole application in a platform-specific way, as this would allow the blog editor to interact with platform-specific data sources.

Annotating a blog entry should be as easy as dragging an address book or calendar entry onto the blog editor. To further support the user in authoring a semantic blog, a visual history of the most recently and most frequently used annotations should be kept. In this way, it will not even be necessary to open the address book or other external applications in order to annotate blog entries with frequently occuring objects. Also, if an index of all desktop objects exist (as it would be in the case of a semantic desktop like Gnowsis or a metadata-enabled file system), the user could be supported with a type-ahead functionality, which suggests matching desktop objects even while the user still types an entry. Taking the example in figure 2, the system could suggest to annotate the entry with matching metadata from the address book (or simply go ahead and do it), as soon as the user types the string "Andreas". Effectively, this would allow a simple kind of (semi-)automatic annotation.

3 The semiBlog Editor

The main focus in the development of semiBlog was to make the authoring of a semantic blog as easy as possible. In particular, we wanted to allow the user to easily add semantic metadata to the blog about entities that already exist on their desktop. This led to the early design decision to make semiBlog a desktop-based application, rather than web-based. As discussed in section 2.2, access to other desktop applications and their data (e.g. through their public APIs), control of the clipboard, and techniques like drag-and-drop are difficult or impossible to implement in a web-based environment. Another design decision was to make the first prototype a native, platform-dependent application — this was done for similar reasons (access to application APIs, etc.), but also because it allowed for a much easier and quicker development process. As a result, semiBlog is currently only available for the Mac OS X operating system. Future versions of semiBlog will possibly be more platform

[9] RDF/XML: `http://www.w3.org/TR/rdf-syntax-grammar/`
[10] FOAF: `http://xmlns.com/foaf/0.1/`

independent, particularly the components comprising the semiBlog Core block and the exporter plugins in figure 4. However, we believe that for an application that wants to interface directly to existing desktop data, a certain degree of platform dependence is always necessary.

3.1 Example Scenario

A screenshot of an example session in semiBlog is shown in figure 3. The user has just had a meeting with his colleague Andreas, where they discussed an academic paper about one of Andreas' projects, named "YARS". He creates a new entry for this in his blog, adds some text and a picture. Then he decides to annotate the entry with some semantic metadata. To do so, he simply selects the name of his colleague, drags the corresponding card from his address book and drops it onto the text area. The entry (or rather a part of it) is now linked to a piece of data on the user's desktop. In a similar fashion he drags and drops the URL of the YARS project page from his web browser, as well as the BibTEX entry for the paper from his bibliography tool. Once a piece of text has been annotated, it is highlighted in the editor, so that each type of data is represented by a different colour. After the blog has been published, other users can read it, and — given their browser supports this — drag the annotations from the blog onto their desktop applications, and thus gain access to the enclosed metadata.

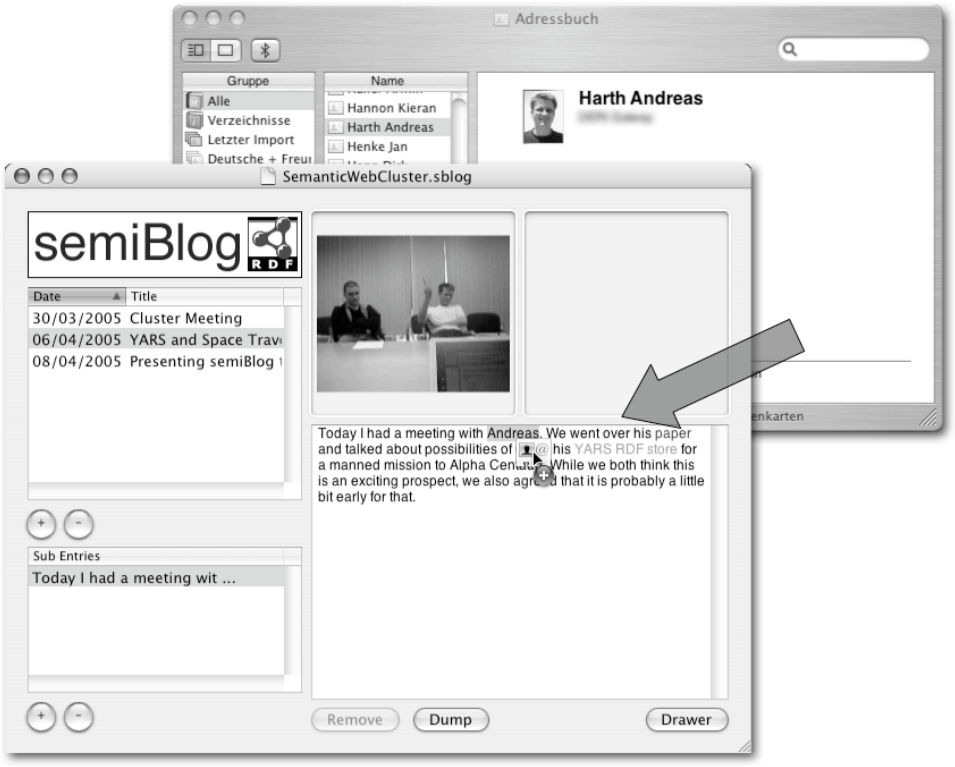


Fig. 3: Annotating a blog with semiBlog

Internally, semiBlog tracks annotations for each individual string within an entry. However, how an annotation is handled once the blog is published on the web,

depends on the publication channel. In the HTML rendering of the entry, the annotation will manifest itself as a simple HTML link to a web page (if such a page can be extracted from the desktop object). In the RDF metadata contained in the newsfeed, the annotation will apply to the entry as a whole, as there is no obvious means to make assertions about specific pieces of text in RDF.

3.2 Architecture and Flow of Data

Figure 4 gives an abstract overview of the architecture and flow of data in semiBlog. The left-hand side of the figure shows some examples of desktop data that a user might have. We show address book entries, bibliographical metadata and web pages, but any other kind of data is conceivable as well. semiBlog's architecture allows custom wrapper plug-ins for each data source, which take the object information from the various desktop applications and transform it into RDF metadata form. Together with the textual entry provided by the user, this metadata is then combined in an intermediate XML representation. To generate the actual blog, the XML is transformed into the various publication channels (currently a non-semantic HTML rendering and the semantic RSS 1.0 feed, which links to the HTML). For the actual publication process, a number of exporter plugins for various kinds of blogging platforms (e.g. Blogger, MovableType, Wordpress, etc) can be chosen. The blog is now ready for publication on the web.

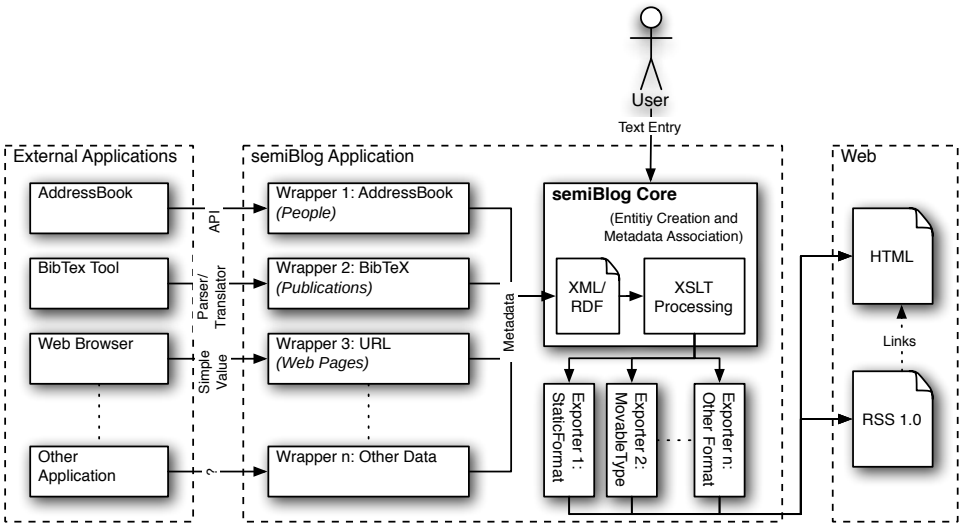


Fig. 4: semiBlog architecture and flow of data

Wrappers For each individual wrapper, access to the data can be handled in a different way: if the data is tied to a specific application (e.g. the AddressBook application for Mac OS X), then access via an application API might be possible. Other, more generic data (e.g. BibTEX) will be handled by appropriate parsers/translators. Wrappers are implemented as plug-ins to the platform, which makes it easy to add new functionality and cover more data sources as the need arises. Plug-ins are basically implementations of an abstract AnnotationType class, which defines the basic API each plug-in has to provide. Parts of this API are getter and setter methods for

the supported data types (objects in a drag-and-drop operation provide their data in various types or data flavors); colours for highlighting in the editor; a method `annotationFromData` to generate RDF; and a method `mainAnchorFromData` to generate a hyperlink reference for the HTML rendering of the blog. Instead of generating a static version of the desktop object in RDF during the annotation process, semiBlog simply stores a URL reference to the object. Each wrapper plugin defines a custom URL scheme for the kind of data it supports. Only when the blog is published at a later stage, will the URL be resolved and an RDF graph be built. This ensures that the metadata contained in the blog is always up-to-date.

Intermediate XML During the publication process, semiBlog uses XML as an intermediate data format, as shown in figure 5, regardless of the exporter plugin chosen. Each `entry` is represented by an entry element, the textual content is contained in a `text` element. This element allows mixed content of text and `annotation` elements, which provides the possibility to add metadata to individual substrings of an entry. We chose this inline annotation technique over external annotation by reference, because it makes the XSLT transformations in the next step easier to accomplish. Each annotation contains a `mainAnchor` to be used in the HTML rendering of the blog (or in any other non-semantic representation), the RDF `metadata` and the actual substring of the entry to be annotated.

```
<!-- ... -->
<text>
  Today I had a meeting with
  <annotation>
    <mainAnchor>http://sw.deri.org/~aharth</mainAnchor>
    <metadata>
      <foaf:Person>
        <foaf:name>Andreas Harth</foaf:name>
        <!-- ... other properties -->
      </foaf:Person>
    </metadata>
    <content>Andreas</content>
  </annotation>.
  We went over his
  <annotation>
    <mainAnchor>
      http://sw.deri.org/2004/06/yars/doc/summary
    </mainAnchor>
    <metadata>
      <bibtex:InProceedings>
        <bibtex:title>
          Yet Another RDF Store:...
        </bibtex:title>
        <!-- ... other properties -->
      </bibtex:InProceedings>
    </metadata>
    <content>paper</content>
  </annotation>
  and talked about possibilities of using his YARS RDF
  store for a manned mission to Alpha Centauri. ...
</text>
<!-- ... -->
```

Fig. 5: Intermediate XML

XSLT Transformation Once the user's textual entry and the semantic metadata have been combined into the intermediate XML format, we can use XSLT to transform it into arbitrary publication channels. This is a rather straight-forward process: for each exporter module, there is one XSLT stylesheet per channel, and

each one picks those elements out of the XML which are appropriate for the corresponding publication channel. For example, the XML ->HTML stylesheet will ignore the `metadata` element and instead pick the `content` and `mainAnchor` tags to produce a hyperlink for specific substrings of the entry. The XML ->RSS stylesheet will only look at the `metadata` element and annotate the entry as a whole. Once this is done, the blog can be uploaded to a server (again depending on the current exporter module).

Extracting Metadata While the functionality to extract metadata from the blog could in principal be added as a plugin to existing blog readers, we chose to build a stand-alone application for demonstration purposes. A user can load a blog's newsfeed an browse its entries. For each entry, the user is presented with a list of annotations, from which he can chose to drag individual items onto his own address book, calendar or other application (see Fig. 6). Technically, the blog reader uses the same plugins as the semiBlog editor, which contain functionality to transform data into both directions (i.e. from the source format into RDF and back). Each plugin defines a pair consisting of an `rdf:type` and a target application, so that the right plugin can be chosen depending on the data dragged and the target of the drag operation.

Fig. 6: A metadata-aware blog reader

4 Related Work

The concept of Semantic Blogging is not our invention; a number of recent papers have investigated the topic from different angles. [5] discusses a semantic blogging prototype built on top of the SW browser Haystack [9]. They interpret blog entries mainly as annotations of other blog entries and web resources in general, and devise a platform to realise this in terms of the SW. We extend the scope of this

interpretation and include the possibility of annotating resources that originally do not exist on the Web, but only on a user's desktop. The paper also underlines the inherent semantic structure of blogs and their entries as such, and presents a way of formalising these semantics in a SW fashion. This point is also made (on a more general level, encompassing various kinds of web based information exchange such as e-mail, bulletin boards, etc.) by [1] and [8]. [3] puts a strong emphasis on the use of semantic technologies to enhance the possibilities of blog consumption, by allowing viewing, navigation and querying with respect to semantics. The paper describes a prototype for both creation and browsing of semantic blogs, which was developed as part of the SWAD-E project . While the prototype only deals with bibliographic metadata as annotations to blog entries, the authors point out that the same technologies can be used for any kind of metadata. [7] describes a platform called Semblog, which uses the FOAF ontology as an integral part. FOAF descriptions of blog authors are linked to their blogs. In this way, the blog as a whole is annotated with metadata about its author. On a more fine-grained level individual blog entries are classified by linking them to a personalised ontology. To implement their platform, the authors provide both a Perl CGI-based tool called RNA and a standalone Windows-based tool called Glucose.

As already mentioned, software like MS Outlook and MeetingMaker allow the transfer of formal data over the web. However, these applications are restricted to specific kinds of data, while our semantic blogging approach aims at covering all kinds of data. Also, the systems typically use proprietary, closed formats, which are aimed specifically at the kind of data they support, while an approach that tries to cover a very broad range of possible kinds of data would have to be open and domain-independent.

Semantic desktop efforts such as Gnowsis [11] assign URLs to every object on a user's desktop, thus allowing to treat them as nodes in an RDF graph and make assertions about them. Also, Gnowsis contains interfaces to extract RDF from various kinds of desktop objects. This approach would offer the perfect infrastructural basis for our semantic blogging efforts, as it would allow us to make our application slimmer and focus on usability ascpects, instead of having to provide functionality for referencing and accessing desktop objects ourselves.

Finally, meta-data enabled file systems like the one implemented in Mac OS X 10.4 ("Tiger"), BeFS or WinFS[11] could be viewed as a lighter version of semantic desktops and could play a similar role as an infrastructural basis. Arbitrary metadata can be added to each file object. By providing a hidden file representation for non-file objects such as entries in the AddressBook application, also non-file objects can benefit from the same functionality. Moreover, since all metadata events are handled at kernel level, changes get noted instantly, and there is little to no chance that changes remain unnoticed, thus compromising the semantic stability of the metadata system.

5 Future Work

Although the full chain of steps involved in publishing a semantic blog writing a textual entry, adding pictures, annotating the entry with semantic metadata and transforming the internal data format into the actual components of the blog is implemented and can be performed within semiBlog, the software is still very much a prototype and can be improved in many areas. Since the main focus was on the interaction between semiBlog and other desktop applications, we have so far neglected aspects such as layout capabilities. As a result, the user is currently bound to a static layout scheme for the blog. In future versions, we will have to address

[11] WinFS: `http://msdn.microsoft.com/data/WinFS/`

this and allow a freer layout, perhaps also offering to the user to choose between different Cascading Style Sheets (CSS) templates. Also, it is currently not possible to add metadata other than that provided by the wrapper plug-ins. Users should have the option to manually annotate entries with arbitrary metadata, if they wish to do so (however, since we were interested in easy drag-and-drop based annotation, this was of no concern to us for the prototype). Also, by using an ontology like SIOC [1], more semantic metadata could be generated easily by making use of the internal and external semantic structure of the blog.

We will also look into adding functionality for automatic or semi-automatic annotation, since even simple and easy manual annotation techniques like drag-and-drop might still be a barrier for many users.

Future versions of the editor will allow integration with generic Semantic Desktop solutions like Gnowsis, or perhaps with metadata-enabled file systems. This would eliminate the need for application specific data wrappers as they are currently used in semiBlog.

6 Conclusion

In this paper, we have argued that semantic blogging can be used as a universal means to perform data exchange, e.g. in companies and organizations (but in no way restricted to these environments). This kind of communication helps to keep up the flow of information in an organization, and can make the work of individual members easier. We have shown that other technologies fail to perform this task. There are certain preconditions that must be met in order to enable semantic blogging in the sense discussed in this paper: we have highlighted the need for a suitable referencing and access mechanism, as well as a suitable metadata exchange format and stressed the importance of usability aspects, such as tight desktop integration and visual UI techniques such as drag-and-drop, to make semantic blogging appealing to non-technical users. Finally, we have presented our semiBlog application, which tries to implement these requirements.

7 Acknowledgement

This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

Bibliography

[1] J. G. Breslin, A. Harth, U. Bojars, and S. Decker. Towards Semantically-Interlinked Online Communities. In *The 2nd European Semantic Web Conference (ESWC '05), Heraklion, Greece, Proceedings, LNCS 3532*, pages 500–514, May 2005.

[2] S. Cayzer. Semantic Blogging and Decentralized Knowledge Management. *Communications of the ACM*, 47(12):47–52, December 2004.

[3] S. Cayzer. Semantic Blogging: Spreading the Semantic Web Meme. In *XML Europe 2004, Amsterdam, Netherlands, Proceedings*, April 2004.

[4] J. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30–37, March/April 2001.

[5] D. R. Karger and D. Quan. What Would It Mean to Blog on the Semantic Web? In *Third International Semantic Web Conference (ISWC2004), Hiroshima, Japan, Proceedings*, pages 214–228. Springer, November 2004.

[6] C. Li. Blogging: Bubble Or Big Deal? Forrester Research, Inc., November 5 2004.

[7] I. Ohmukai and H. Takeda. Semblog: Personal Publishing Platform with RSS and FOAF. In *1st Workshop on Friend of a Friend, Social Networking and the (Semantic) Web, Proceedings*, pages 217–221, Galway, September 2004.

[8] D. Quan, K. Bakshi, and D. R. Karger. A Unified Abstraction for Messaging on the Semantic Web. In *The Twelfth International World Wide Web Conference (WWW2003), Budapest, Hungary, Proceedings*, May 2003.

[9] D. Quan, D. Huynh, and D. R. Karger. Haystack: a Platform for Authoring End User Semantic Web Applications. In *Second International Semantic Web Conference (ISWC2003), Proceedings*, 2003.

[10] M. Röll. Business Weblogs — A Pragmatic Approach to Introducing Weblogs in Medium and Large Enterprises. In *BlogTalk - A Eurpean Conference on Weblogs, Vienna, Austria*, May 2003.

[11] L. Sauermann. The Gnowsis — Using Semantic Web Technologies to build a Semantic Desktop. Master's thesis, Technische Universität Wien, December 2003.

[12] B. Stone. *Blogging - Genius Strategies for Instant Web Content.* New Riders, 2002.

[13] J. Walker. Weblog. In D. Herman, M. Jahn, and M.-L. Ryan, editors, *Routledge Encyclopedia of Narrative Theory*, page 45. Routledge, London and New York, 2005.

Semantically Enhanced Searching and Ranking on the Desktop

Paul - Alexandru Chirita, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu

L3S Research Center / University of Hanover
Deutscher Pavillon, Expo Plaza 1
30539 Hanover, Germany
{chirita,ghita,nejdl,paiu}@l3s.de

Abstract. Existing desktop search applications, trying to keep up with the rapidly increasing storage capacities of our hard disks, offer an incomplete solution for the information retrieval. In this paper we describe our desktop search prototype, which enhances conventional full-text search with semantics and ranking modules. In this prototype we extract and store activity-based metadata explicitly as RDF annotations. Our main contributions are represented by the extensions we integrate into the Beagle desktop search infrastructure to exploit this additional contextual information for searching and ranking the resources on the desktop. Contextual information plus ranking brings desktop search much closer to the performance of web search engines. The initially disconnected sets of resources on the desktop are connected by our contextual metadata, and then a PageRank derived algorithm allows us to rank these resources appropriately. Finally, we use a detailed working scenario to discuss the advantages of this approach, as well as the user interfaces of our search prototype.

1 Introduction

The capacity of our hard-disk drives has increased tremendously over the past decade, and so has the number of files we usually store on our computer. It is no wonder that sometimes we cannot find a document any more, even when we know we saved it somewhere. Ironically, in quite a few of these cases nowadays, the document we are looking for can be found faster on the World Wide Web than on our personal computer.

Web search has become more efficient than PC search due to the boom of web search engines and due to powerful ranking algorithms like the PageRank algorithm introduced by Google [10]. The recent arrival of desktop search applications, which index all data on a PC, promises to increase search efficiency on the desktop. However, even with these tools, searching through our (relatively small set of) personal documents is currently inferior to searching the (rather vast set of) documents on the web. This happens because these desktop search applications cannot rely on PageRank-like ranking mechanisms, and they also fall short of utilizing desktop specific characteristics, especially context information.

It is thus obvious that simple indexing of the data on the desktop has to be enhanced by ranking techniques, otherwise the user has no other choice but to look at the entire result sets for her queries – usually a tedious task. The main problem with ranking on the desktop comes from the lack of links between documents, the foundation of current ranking algorithms (in addition to TF/IDF numbers). A semantic desktop offers the missing ingredients: By gathering semantic information from user activities, from the contexts the user works in[1], we build the necessary links between documents.

In this paper we enhance and contextualize desktop search based on both resource specific metadata and semantic metadata collected from different available contexts and activities performed on a personal computer. We first describe the semantics of these different contexts by appropriate ontologies in Section 3.3, and then propose a ranking algorithm for desktop documents in Section 3.4. For this latter aspect, we focus on recent advances of PageRank-based ranking, showing how local (i.e., context-based) and global ranking measures can be integrated in such an environment. We are currently implementing our prototype on top of the open source Beagle project [8], which aims to provide sophisticated desktop search in Linux. Section 4 gives a detailed description of our prototype, and shows how we extended Beagle with additional modules for annotating and ranking resources. In Section 5 we use one typical search scenario in order to illustrate how our extended Beagle search infrastructure can improve the retrieval of search results in terms of number and order of results, using contextual information and ranking based on this information.

2 Previous Work

The difficulty of accessing information on our computers has prompted several first releases of desktop search applications during the last months. The most prominent examples include Google desktop search [9] (proprietary, for Windows) and the Beagle open source project for Linux [8]. Yet they include *no* metadata whatsoever in their system, but just a regular text-based index. Nor does their competitor MSN Desktop Search [11]. Finally, Apple Inc. promises to integrate an advanced desktop search application (named *Spotlight Search* [2]) into their upcoming operating system, Mac OS Tiger. Even though they also intend to add semantics into their tool, only explicit information is used, such as file size, creator, last modification date, or metadata embedded into specific files (images taken with digital cameras for example include many additional characteristics, such as exposure information or whether a flash was used). While this is indeed an improvement over regular search, it still misses contextual information often resulting or inferable from explicit user actions or additional background knowledge, as discussed in the next sections.

[1] Studies have shown that people tend to associate things to certain contexts [14], and this information should be utilized during search. So far, however, neither has this information been collected, nor have there been attempts to use it.

Swoogle [5] is a search and retrieval system for finding semantic web documents on the web. The ranking scheme used in Swoogle uses weights for the different types of relations between Semantic Web Documents (SWD) to model their probability to be explored. However, this mainly serves for ranking between ontologies or instances of ontologies. In our approach we have instances of a fixed ontology and the weights for the links model the user's preferences.

Facilitating search for information the user has already seen before is also the main goal of the *Stuff I've Seen (SIS)* system, presented in [6]. Based on the fact that the user has already seen the information, contextual cues such as time, author, thumbnails and previews can be used to search for and present information. [6] mainly focuses on experiments investigating the general usefulness of this approach though, without presenting more technical details.

The importance of semantically capturing users' interests is analyzed in [1]. The purpose of their research is to develop a ranking technique for the large number of possible semantic associations between the entities of interest for a specific query. They define an ontology for describing the user interest and use this information to compute weights for the links among the semantic entities. In our system, the user's interest is a consequence of her activities, this information is encapsulated in the properties of the entities defined, and the weights for the links are manually defined.

An interesting technique for ranking the results for a query on the semantic web takes into consideration the inferencing processes that led to each result [13]. In this approach, the relevance of the returned results for a query is computed based upon the specificity of the relations (links) used when extracting information from the knowledge base. The calculation of the relevance is however a problem-sensitive decision, and therefore task oriented strategies should be developed for this computation.

3 An Architecture for Searching the Semantic Desktop

3.1 Overview

This chapter will present our 3-layer architecture for generating and exploiting the metadata enhancing desktop resources. At the bottom level, we have the physical resources currently available on the PC desktop. Even though they can all eventually be reduced to "files", it is important to differentiate between them based on content and usage context. Thus, we distinguish structured documents, emails, offline web pages, general files[2] and file hierarchies. Furthermore, while all of them do provide a basis for desktop search, they also miss a lot of contextual information, such as the author of an email or the browsing path followed on a specific web site. We generate and store this additional search input using RDF metadata, which is placed on the second conceptual layer of our architecture. Finally, the uppermost layer implements a ranking mechanism over all resources from the previous levels. An importance score is thus computed for

[2] Text files or files whose textual content can be retrieved.

each desktop item, supporting an enhanced ordering of results within desktop search applications. The entire architecture is depicted in Figure 1. In the next subsections we will describe each of its layers following a bottom-up approach.

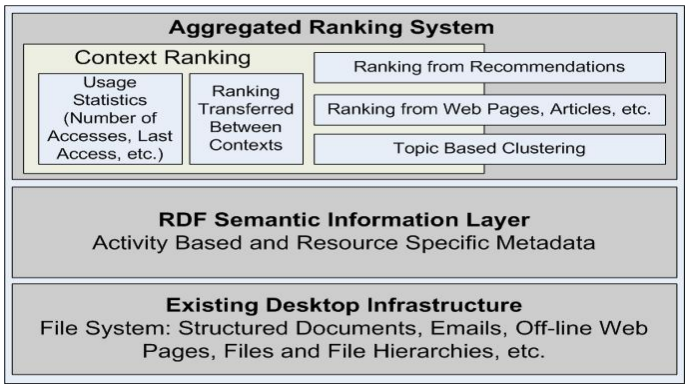


Fig. 1. Desktop Ranking System Architecture

3.2 Current Desktop Infrastructure and its Limitations

Motivation and Overview. Today the number of files on our desktops can easily reach 10,000, 100,000 or more. This large amount of data can no longer be ordered with manual operations such as defining explicit file and directory names. Automatic solutions are needed, preferably taking into account the activity contexts under which each resource was stored/used. In our prototype we focus on three main working contexts of email exchanges, file procedures (i.e., create, modify, store, etc.), and web surfing. Furthermore, we investigate an additional extended context related to research and scientific publications. In the following paragraphs, we will discuss how the resources associated to these contexts are currently encountered, and which (valuable) information is lost during their utilization. Subsequent sections will then present solutions to represent this information and exploit it for desktop search applications.

Email Context. One of the most flourishing communication mediums is certainly email communication. For example, international scientific collaboration has become almost unthinkable without electronic mail: Outcomes of brainstorming sessions, intermediate versions of reports, or published articles represent just a few of the items exchanged within this environment. Similarly, Internet purchasing or reservations are usually confirmed via email. However, if we consider the continuous increase of email exchanges, other enhanced solutions will be necessary to sort our correspondence. More, when storing emails, a lot of contextual information is lost. Most significant here is the semantic connection between the attachments of an email, its sender and subject information, as well

as the valuable comments inside its body. We propose to explicitly represent this information as RDF metadata, to enable both enhanced search capabilities for our inbox, as well as the exploitation of the semantic link between desktop files (e.g., PDF articles stored from attachments), the person that sent them to us via email or the comments she added in the email body.

Files and File Hierarchy Context and Web Cache Context. Due to space limitations, we refer the reader to [4], where we proposed several solutions to enrich the information associated to file and directory names, as well as to the previously visited resources on the Web.

Working with Scientific Publications. Research activities represent one of the occupations where the need for contextual metadata is very high. The most illustrative example is the publication itself: Where did this file come from? Did we download it from CiteSeer or did somebody send it to us by email? Which other papers did we download or discussed via email at that time, and how good are they (based on a ranking measure or on their number of citations)? We might remember the general topic of a paper and the person with whom we discussed about it, but not its title. These questions arise rather often in a research environment and have to be answered by an appropriate search infrastructure.

3.3 RDF Semantic Information Layer

Motivation and Overview. People organize their lives according to preferences often based on their activities. Consequently, desktop resources are also organized according to performed activities and personal profiles. Since, as described above, most the information related to these activities is lost on our current desktops, the goal of the RDF semantic information layer is to record and represent this data and to store it in RDF annotations associated to each resource. Figure 2 depicts an overview image of the ontology that defines appropriate annotation metadata for the context we are focusing on in this paper. The following paragraphs will describe these metadata in more detail.

Email Metadata. Basic properties for the email context are referring to the date when an email was sent or accessed, as well as its subject and body text. The status of an email can be described as seen/unseen or read/unread. A property "reply to" represents email thread information, the "has attachment" property describes a 1:n relation between an email and its attachments. The "sender" property gives information about the email sender, which can be associated to a social networking trust scheme, thus providing valuable input for assessing the quality of the email according to the reputation of its sender.

File and Web Cache Specific Metadata. For these, we again refer the reader to our previous work [4] describing the ontologies associated to these two activity contexts. An overview can be found in the lower half of Figure 2.

Scientific Publications Metadata. The Publication class represents a specific type of file, with additional information associated to it. The most common fields are "author", "conference", "year", and "title", which comprise the regular information describing a scientific article. Additionally, we store the paper's CiteSeer ID (if any). The publication context is connected to the email context,

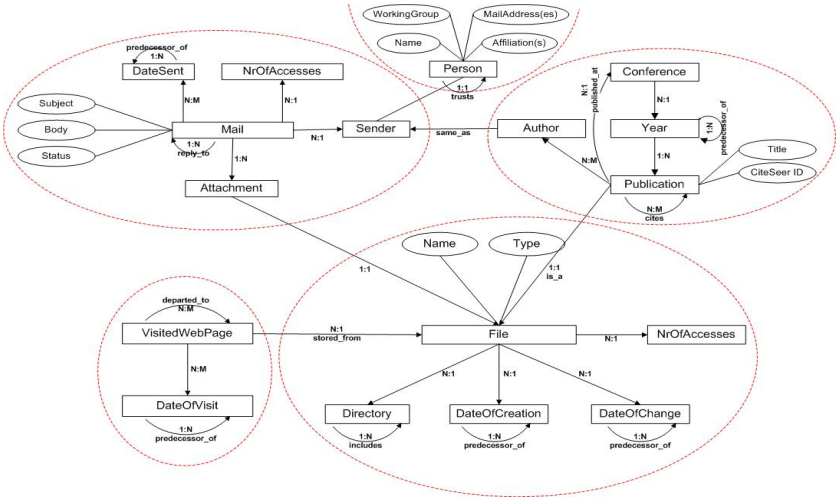


Fig. 2. Contextual Ontology for the Semantic Desktop

if we communicate with an author or if we save a publication from an email attachment. Of course, since each publication is stored as a file, it is also connected to the file context, and thus to the file specific information associated to it (e.g., path, number of accesses, etc.).

3.4 Aggregated Ranking System

Motivation and Overview. As the amount of desktop items has been increasing significantly over the past years, desktop search applications will return more and more hits to our queries. Contextual metadata, which provide additional information about each resource, result in even more search results. A measure of importance is therefore necessary, which enables us to rank these results. The following paragraphs describe such a ranking mechanism, developed based on the Google PageRank algorithm [10].

Basic Ranking. Given the fact that rank computation on the desktop would not be possible without the contextual information, which recreates the links among resources, annotation ontologies should describe all aspects and relationships among resources influencing the ranking. The identity of the authors for example influences our opinion of documents, and thus "author" should be represented explicitly as a class in our publication ontology.

Second, we have to specify how these aspects influence importance. ObjectRank [3] has introduced the notion of authority transfer schema graphs, which extend schemas similar to the ontologies previously described, by adding weights and edges in order to express how importance propagates among the entities and resources inside the ontology. These weights and edges represent authority transfer annotations, which extend our context ontologies with the information

we need to compute ranks for all instances of the classes defined in the context ontologies.

Figure 3 depicts our context ontology plus appropriate authority transfer annotations. For example, authority of an email is split among the sender of the email, its attachment, the number of times that email was accessed, the date when it was sent and the email to which it was replied. If an email is important, the sender might be an important person, the attachment an important one and/or the number of times the email was accessed is very high. Additionally, the date when the email was sent and the previous email in the thread hierarchy also become important. As suggested in [3], every edge from the schema graph is split into two edges, one for each direction. This is motivated by the observation that authority potentially flows in both directions and not only in the direction that appears in the schema: if we know that a particular person is important, we also want to have all emails we receive from this person ranked higher. The final ObjectRank value for each resource is calculated based on the PageRank formula (presented in Section 4.3).

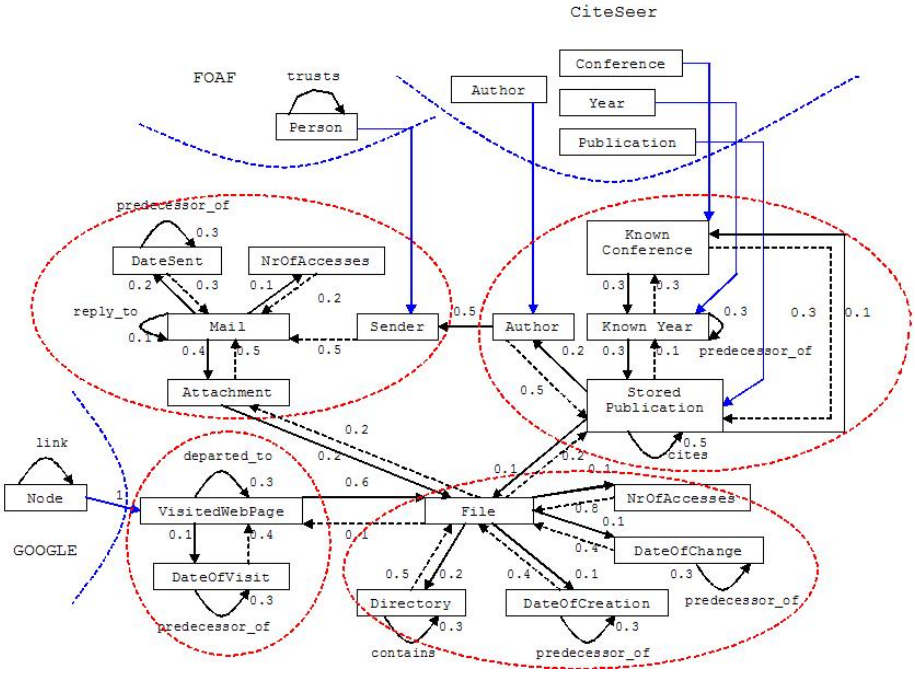


Fig. 3. Contextual Authority Transfer Schema

Using External Sources. For the computation of authority transfer, we can also include additional external ranking sources to connect global ranking computation and personalized ranking of resources on our desktop. These exter-

nal ranking sources are used to provide the seed values for the calculation of the personal ranking. Our prototype ontology includes three global ranking services, one returning Google ranks, the second one ranks computed from the CiteSeer database and the last one from the social network described with FOAF.

The ObjectRank value for each resource is calculated based on the PageRank formula and the seed values for this computation integrate information from external ranking systems and personalized information. We use the following external ranking systems as the most relevant for our purpose:

- *Ranking for articles.* Co-citation analysis is used to compute a primary rank for the article [12]. Because of the sparse article graph on each desktop this rank should be retrieved from a server that stores the articles (in our case all metadata from CiteSeer and DBLP).
- *Recommendations.* We may receive documents from other peers together with their recommendations. These recommendations are weighted by a local estimate of the sender's expertise in the topic [7].

Personalization. Different authority transfer weights express different preferences of the user, translating into personalized ranking. The important requirement for doing this successfully is that we include in a users ontology all concepts, which influence her ranking function. For example, if we consider a publication important because it was written by an author important to us, we have to represent that in our context ontology. Another example are digital photographies, whose importance is usually heavily influenced by the event or the location where they were taken. In this case both event and location have to be included as classes in our context ontology. The user activities that influence the ranking computation have also to be taken into account, which translates to assigning different weights to different contexts.

4 Prototype

Our current prototype is being built on top of the open source Beagle desktop search infrastructure, which we extended with additional modules: metadata generators, which handle the creation of contextual information around the resources on the desktop, and a ranking module, which computes the ratings of resources so that search results are shown in the order of their importance. The advantage of our system over existing desktop search applications consists in both the ability of identifying resources based on an extended set of attributes – more results, and of presenting the results according to their ranking – to enable the user to quickly locate the most relevant resource.

4.1 Current Beagle Architecture

The main characteristic of our extended desktop search architecture is metadata generation and indexing on-the-fly, triggered by modification events generated

upon occurrence of file system changes. This relies on notification functionalities provided by the kernel. Events are generated whenever a new file is copied to hard disk or stored by the web browser, when a file is deleted or modified, when a new email is read, etc. Much of this basic notification functionality is provided on Linux by an inotify-enabled Linux kernel, which is used by Beagle.

Our prototype keeps all the basic structure of Beagle and adds additional modules that are responsible for generating and using the contextual annotations enriching the resources on our desktop. The main components of the extended Beagle prototype are Beagled^{++} and Best^{++}, as seen in Figure 4, "++" being used to denote our extensions. Beagled^{++} is the main module that deals with indexing of resources on the desktop and also retrieving the results from user queries. Best^{++} is responsible for the graphical interface, put on top of Beagled^{++}, communicating through the DBus daemon. Whenever a DBus daemon launches, a new bus is initialized and to this one both Beagled^{++} and Best^{++} connect and register. Then Beagled^{++} starts a global scheduler thread and the inotify thread, which are responsible for the indexing of new resources found on the desktop. The query driver is then started and waits for queries. The Best^{++} interface is also initialized, using this module queries will be transmitted to Beagled^{++} and answers visualized.

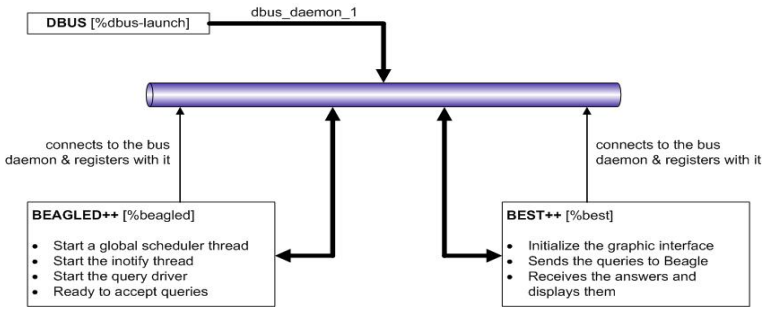


Fig. 4. Extended Beagle Desktop Search

4.2 Extending Beagle with Metadata Generators

Depending on the type and context of the file / event, metadata generation is performed by appropriate metadata generators, as described in Figure 5. These applications build upon an appropriate RDFS ontology as shown in [4], describing the RDF metadata to be used for that specific context. Generated metadata are either extracted directly (e.g. email sender, subject, body) or are generated using the appropriate association rules plus possibly some additional background knowledge. All of these metadata are exported in RDF format, and added to a metadata index, which is used by the search application together with the usual full-text index.

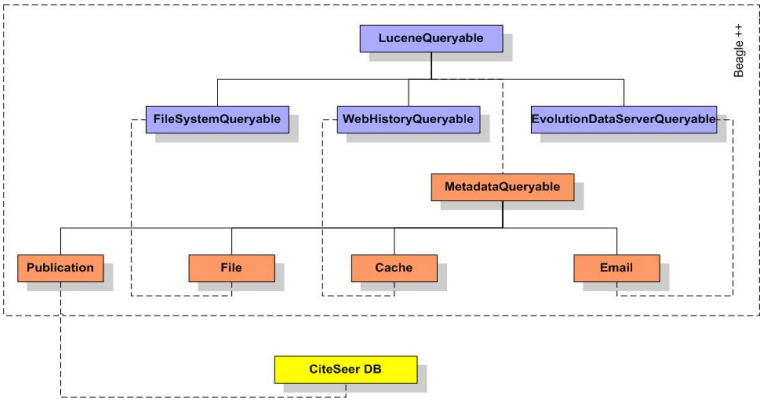


Fig. 5. Beagle Extensions for Metadata Support

The architecture of our prototype environment includes four prototype metadata generators according to the types of contexts described in the previous sections. We added a new subclass of the LuceneQueryable class, MetadataQueryable, and, from this one, derived four additional subclasses, dealing with the generation of metadata for the appropriate contexts (Files, Web Cache, Emails and Publications). The annotations we create include the corresponding elements depicted in the ontology graph Figure 2. They are described in detail in [4]. A new one is the publication metadata generator, described in the next paragraph.

Publication Metadata Generator. For the experiments described in this paper, we have implemented a metadata generator module, which deals with publications. For each identified paper, it extracts the title and tries to match it with an entry into the CiteSeer publications database. If it finds an entry, the application builds up an RDF annotation file, containing information from the database about the title of the paper, the authors, publication year, conference, papers which cite this one and other CiteSeer references to publications. All annotation files corresponding to papers are merged in order to construct the RDF graph of publications existing on one's desktop.

4.3 Extending Beagle with A Ranking Module

Each user has his own contextual network / context metadata graph and for each node in this network the appropriate ranking as computed by the algorithm described in section 3.4. The computation of rankings is based on the link structure of the resources as specified by the defined ontologies and the corresponding metadata. We base our rank computation on the PageRank formula

$$r = d \cdot A \cdot r + (1 - d) \cdot e \tag{1}$$

applying the random surfer model and including all nodes in the base set. The random jump to an arbitrary resource from the data graph is modeled by the

vector e. A is the adjacency matrix which connects all available instances of the existing context ontology on one's desktop. The weights of the links between the instances correspond to the weights specified in the authority transfer annotation ontology. Thus, when instantiating the authority transfer annotation ontology for the resources existing on the users desktop, the corresponding matrix A will have elements which can be either 0, if there is no edge between the corresponding entities in the data graph, or they have the value of the weight assigned to the edge determined by these entities, in the authority transfer annotation ontology. Additionally, in the case of publications, for the rank computation we also take into account the ratings extracted from the CiteSeer database, with the aid of our publication metadata generator. This values are used as seed values for the calculation of the personalized rankings.

5 In-Depth Scenario and Results

We have considered a set of test scenarios for validating our assumptions about the benefits of our metadata enhanced search engine. Let us look at one of them in more detail. We assume that Bob and Alice are two team members of a computer science research institute, both being interested in semantic web technologies. Alice is currently writing a paper about searching and ranking on the semantic desktop and she wants to find some good papers on this topic, which she remembers she stored sometime ago on her desktop. Figure 6 depicts a small part of the set of publications existing on Alice's desktop, along with the contextual metadata associated with them. Papers are connected among each others by the 'cites'- relationship. By dotted lines we represent the papers that are not stored on Alice's desktop, but are referenced by some of the saved ones. In order to find them she makes use of the extended desktop search engine and issues a query with the searched terms **'semantic desktop'**. Let us look at the results she will find.

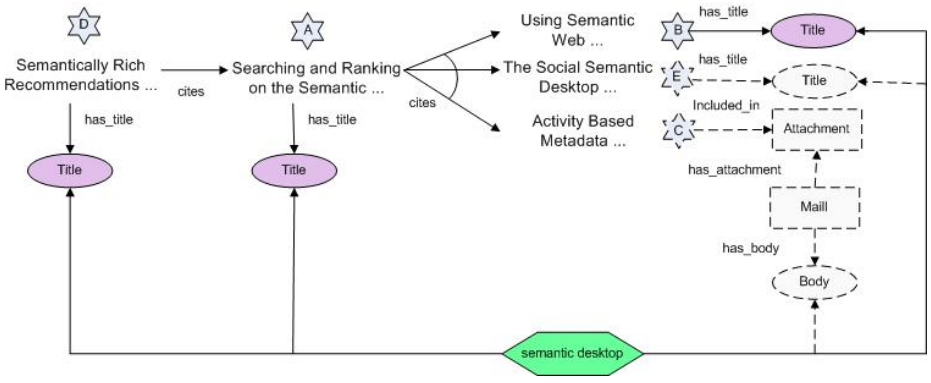


Fig. 6. Papers example

The next three subsections discuss this scenario in more detail. We describe the type of hits our extended Beagle engine returns, show how we display these results together with their additional metadata and how ranking information is used to order the search results.

5.1 Direct Hits vs. Metadata Hits

The hits returned by any normal search engine (direct hits) would be the ones that contain the searched keywords in the title or in the content of the publication, in our case publications **A**, **B** and **D** ("Searching and Ranking on the **Semantic Desktop**", "Using Semantic Web Technologies to Build a **Semantic Desktop**", "Semantically Rich Recommendations in Social Networks for Sharing and Exchanging Semantic Context"). Traditional search engines would not be able to retrieve the other two publications. Publication **C** is included as email attachment, and not indexed by Beagle. However, the corresponding email text contains the searched keywords, and therefore we can retrieve it as an indirect hit (from the metadata as for each email we automatically store the email text as metadata for all attachments). Publication **E** is not even stored on the computer but is among the cited publications by another stored publication(**A**) which contains the keywords in the title, and therefore will be returned as a metadata hit("The Social **Semantic Desktop**").

We extend the Best interface provided by Beagle to include the metadata hits together with the direct ones. The direct hits are shown as Beagle normally does, with the occurrences of the searched terms emphasized. For the indirect hits we display the resources whose associated metadata include the query terms. Our example shows the first 5 out of a total of 20 results. As depicted in Figure 7, the first hit is an email having as attachment a publication not stored on the computer but refers to the topic in the body of the email, which contains the searched keywords. The last result in this picture is another indirect hit but the resource that is displayed is neither explicitly nor implicitly (e.g. in the email attachment) stored on the computer. This is why the user is redirected to the results provided by Google when searching for the item that it is pointed to by the metadata file.

5.2 Rankings

Results. As Figure 7 shows, the results are displayed according to the computed resource rankings. The first hit has the highest rank among the resources in the result set. The rank values are also shown so that the user has an impression about how relevant the results are. In our case the most important result is the email Alice received from Bob including in its attachment a publication. Alice has exchanged her context with Bob, as well as with Caroline, Dan and Tom. All these persons have a high level of trust for Bob and therefore, by exchanging context information, in the resulting graph the node corresponding to Bob will have many incoming links. This translates into a high rank value for Bob. As we suggested in Figure 3, certain percentages of Bob's rank will flow towards all

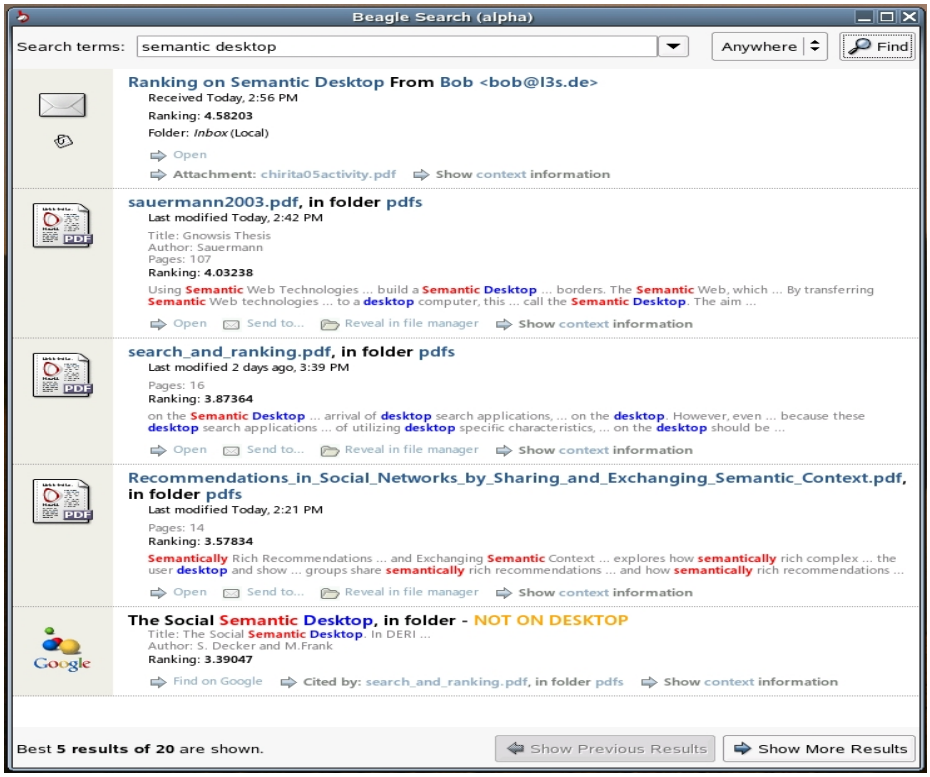


Fig. 7. Beagle^{++} Main Window

nodes that Bob's node points to. Since Bob is the sender of the email which was identified by Beagle as a match for the searched terms, the rank of this hit has a very high value.

We additionally observe that the last hit from this partial set of results, in spite of the fact that is not stored on the desktop, has also a high rank because it is cited by the third hit of this query. Its high rank is also influenced by other resources that the publication receives links from.

Discussion. Ranking algorithms are very much influenced by the structure of their underlying graph. More specifically, different distributions of links yield rather different resource orderings. Luckily, the cached web pages and the scientific publications do exhibit a web-like link distribution (i.e., power-law), and are thus suitable for such a ranking scheme as ours. However, even though in our initial experiments from [4] and from this paper we found the desktop resource graphs to show an encouraging structure (also when emails and regular files have been included), we are currently validating these results on a larger scale of users and search scenarios.

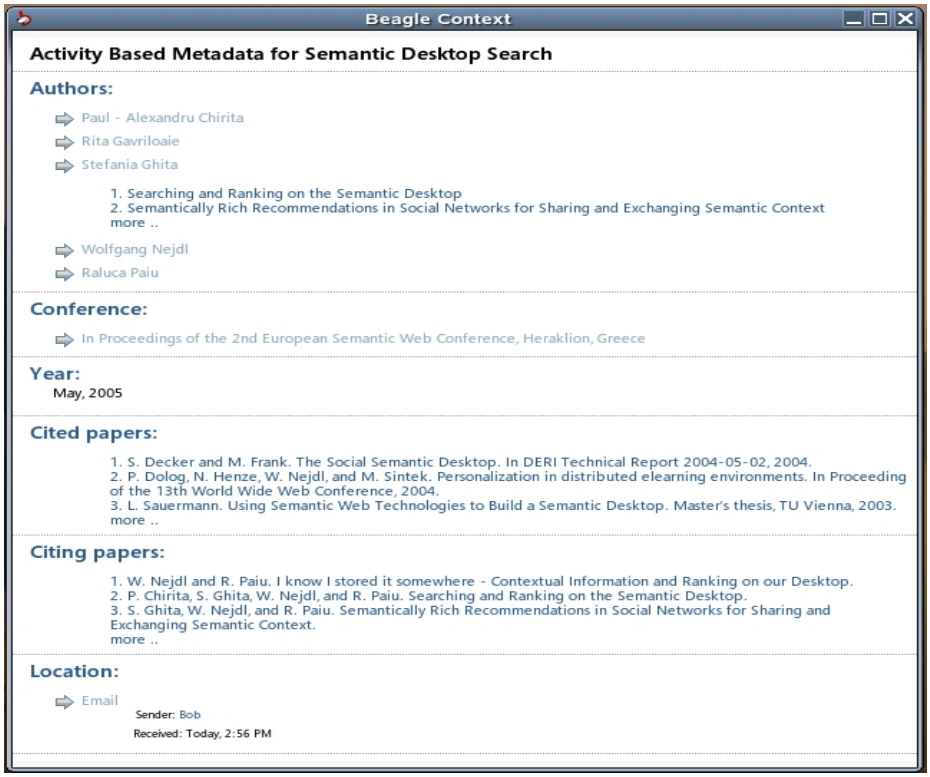


Fig. 8. Beagle^{++} Metadata Window

5.3 Metadata Visualization

Whenever a user clicks on the 'Show context information' link for a certain result, the corresponding metadata can be visualized, both for direct or indirect hits. A new window pops up displaying a list of details that correspond to the ontology related to the type of resource. Alice chooses to visualize the first result returned by Beagle, representing an email from Bob and having as attachment a publication. Since the interesting resource for this query is the PDF file in the attachment, the metadata window displays the annotations corresponding to publications together with other contextual information associated with it. The publication "Activity Based Metadata for Semantic Desktop Search" has 5 authors and for each of the authors we can further display the next level of metadata. For example, in Figure 8, Alice extended author S. Ghita and she can see other publications of this author. Additionally, she can see that the publication was presented at the ESWC conference in 2005, its referenced publications and the ones that cited it. Information related to the provenance of this resource is also shown, the email it was saved from and its sender.

6 Conclusions and Future Work

We presented two main contributions that enhance traditional desktop search, focusing on how regular text-based desktop search can be enhanced with semantics / contextual information and ranking exploiting that information. Searching for resources then will not only retrieve explicit results but also items inferred from the users' existing network of resources and contextual information. Maintaining the provenance of information can help the search engine take into account the recommendations from other users and thus provide more retrieved results. The ranking module, by exploiting contextual information, improves retrieval and presentation of search results, providing more functionality to desktop search.

There are quite a few interesting additional contexts, that are worth investigating in the future: metadata embedded in multimedia files, the relations between objects embedded within each other (a presentation including pictures, tables, charts, etc.), or chat history. A further interesting question we want to investigate in the future is how to learn contextual authority transfer weights from user feedback on ranked search results.

References

1. B. Aleman-Meza, C. Halaschek, I. B. Arpinar, and A. Sheth. Context-aware semantic association ranking. In *Semantic Web and Databases Workshop*, 2003.
2. Apple spotlight search. http://developer.apple.com/macosx/tiger/spotlight.html.
3. A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, Toronto, Sept. 2004.
4. P. A. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. In *In Proceedings of the 2nd European Semantic Web Conference*, Heraklion, Greece, May 2005.
5. L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs. Swoogle: A search and metadata engine for the semantic web. In *in Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*, Washington, DC, Nov. 2004.
6. S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *Proc. of the 26th Intl. ACM SIGIR Conference*, Toronto, July 2003.
7. S. Ghita, W. Nejdl, and R. Paiu. Semantically rich recommendations in social networks for sharing and exchanging semantic context. In *ESWC Workshop on Ontologies in P2P Communities*, 2005.
8. Gnome beagle desktop search. http://www.gnome.org/projects/beagle/.
9. Google desktop search application. http://desktop.google.com/.
10. Google search engine. http://www.google.com.
11. Msn desktop search application. http://beta.toolbar.msn.com/.
12. A. Sidiropoulos and Y. Manolopoulos. A new perspective to automatically rank scientific conferences using digital libraries. In *Information Processing and Management 41 (2005) 289Z12*, 2005.
13. N. Stojanovic, R. Studer, and L. Stojanovic. An approach for the ranking of query results in the semantic web. In *ISWC*, 2003.
14. J. Teevan, C. Alvarado, M. Ackerman, and D. R. Karger. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *CHI*, 2004.

SemperWiki: a semantic personal Wiki

Eyal Oren
eyal.oren@deri.org

Digital Enterprise Research Institute
Galway, Ireland

Abstract. Wikis are collaborative authoring environments, and are very popular. The original concept has recently been extended in two directions: semantic Wikis and personal Wikis. Semantic Wikis focus on better retrieval and querying facilities, by using semantic annotations of pages. Personal Wikis focus on improving usability and on providing an easy-to-use personal information space. We combine these two developments and present a semantic personal Wiki. Our application SemperWiki offers the usability of personal Wikis and the improved retrieval and querying of semantic Wikis. Users can annotate pages with RDF together with their normal text.The system is extremely easy-to-use, provides intelligent navigation based on semantic annotations, and responds instantly to all changes.

1 Introduction

The amount of information we process, maintain, search, and use in our daily work is enormous. All this information is contained in our *desktop*, our personal working space. Efficiently and effectively retrieving the relevant information from our desktop is currently a problem, especially since many related pieces of information are spread over various applications that do not communicate with each other (e.g. some emails, documents, appointments, and websites on the same topic).

Adding semantic annotations to desktop data could alleviate this problem and offer better retrieval possibilities [2, 5]. By annotating data on the desktop in a semantic language such as RDF we get (i) a uniform data format for the separate pieces of information, (ii) integration of information pieces based on URIs, (iii) enhancement of information with background knowledge in ontologies.

Semantic annotations add application-independent meaning and structure to data; they allow better retrieval of our desktop data. To some extent, these annotations can be provided by special application wrappers that know the structure and semantics of some particular application data and can export this in RDF. But the user will also have to make manual annotations, because the semantics of some information chunk is not always available or derivable from the application data.

For example, a wrapper can export meeting appointments from a calendar application into RDF, but if the user wants to annotate the participants, topic,

discussion, and outcome of the meeting, he will have to add these annotations manually because these data are not available in the calendaring application. Also, if the user wants to relate the meeting agenda and the meeting notes to the meeting appointment, he will have to add these annotations manually (because again, the application wrapper cannot make these relations automatically).

But adding semantic annotations to information pieces requires effort. Users will only make this effort if they benefit from it. Manually annotating data should therefore be easy and rewarding, since users will otherwise not annotate their desktop data. If we want the user to add annotations, we need to entice him to do so.

We present a desktop application that does exactly this: it entices users to create semantic data. This semantic personal Wiki (SemperWiki) serves as a personal information system. It is extremely easy to use and provides instant gratification for adding semantic annotations.

SemperWiki is a best-of-breed between semantic Wikis and personal Wikis, two orthogonal extensions to the original Wiki concept. Personal Wikis focus on extreme ease of use; semantic Wikis focus on improved retrieval and navigation. Combining them offers the usability of personal Wikis and the gratification of semantic Wikis.

SemperWiki is not limited to the annotation of desktop data; it is a general-purpose tool for creating and using semantically annotated data that addresses a basic prerequisite towards a better desktop: helping and enticing users to add semantic annotations. It can be used as a stand-alone semantic information system or as user interface of a semantic desktop system.

The paper is structured as follows: first we shortly recall the original idea of a Wiki in section 2. Then we review the characteristics of semantic Wikis in section 3 and personal Wikis in section 4, and we analyse the functionality of several Wiki systems in section 5. We introduce the concept of semantic personal Wikis and the SemperWiki application in section 6, and conclude with points for future work.

2 Wikis

Wiki Wiki Webs were first introduced by Leuf and Cunningham [4]. Wikis are interlinked web sites that can be collaboratively edited by anyone. Pages are written in a simple syntax so that even novice users can easily edit pages. The syntax consists of simple tags for creating links to other Wikipages and textual markups such as lists and headings.

The user interface of most Wikis consists of two modes: in reading mode, the user is presented normal webpages that can contain pictures, links, textual markup, etc. In editing mode, the user is presented an editing box displaying the Wiki syntax of the page (containing the text including the markup tags). During editing, the user can request a preview of the page, which is then rendered by the server and returned to the user.

Many Wiki engines exist for anyone who wants to setup a Wiki, most of these engines are open-source. Many sites run a Wiki as a community venue, enabling users to discuss and write on topics. For example, many open-source projects have a documentation Wiki, where users can collaboratively add documentation about the project. The burden of editing is thus shared over the whole community, while still allowing anybody to quickly find relevant documentation (which is harder in e.g. a forum or bulletin board). Popular Wikis such as Wikipedia[1] can grow very fast, since interested visitors can edit and create pages at will.

Wikis are inherently server-based which has the advantage of user platform independence (users only need a browser) but the disadvantage of requiring a round-trip for all changes[2]. For example, after each edit the page is committed back to the server, and the newly rendered page is returned to the user. Although the usability of most Wikis is good compared to other web applications (being simple and fast), the need for server round-trips is a disadvantage compared to desktop applications.

A problem with large Wikis is finding relevant information. Since almost all the information in current Wikis is textual, the only possibility to locate relevant information is a full-text search on some keywords. The only semantics of pages lies in the links between pages. Indeed almost all Wiki engines generate navigational benefits from these links: one can see all pages linking to the current one, and go to these related pages. But this navigation through related pages is quite limited, and does not address the need for more intelligent information retrieval.

3 Semantic Wikis

Several systems have been developed that enable users of Wikis to semantically annotate the information they publish in the Wiki. They enable users to structure and annotate the content of pages; they reward the user for doing so by adding navigational possibilities to these annotated pages.

3.1 Examples

We list some examples of semantic Wikis. We discuss only those systems that are being actively developed or that have been published about.

Platypus Platypus Wiki is a Wiki that is augmented with semantic technologies [7]. Users can annotate information about pages by constructing RDF triples of the form (page, predicate, object). The subject of the triple is always the current page, the predicate and object can be resources.

In Platypus the editing interface consists of three pane: one pane contains the page text in Wiki syntax, the other two contain RDF statements about the

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

[2] approaches exist to enhance user experience by communicating in the background, but these do still not compare to the usability possibilities of a desktop application.

page. Writing text on a page and writing semantic annotations are therefore distinct activities in the Wiki, and the user has to consciously switch between editing normal text and editing semantic annotations.

Platypus offers a query interface, separated from the normal page view. The query possibilities are rich, and the background knowledge in the ontologies is used in retrieving all answers to the queries. This inferencing capability means that users can retrieve information that was not added explicitly but derived by the system.

Platypus Wiki is an initial attempt to augment Wikis with semantics, all pages in the Wiki can be annotated in RDF, and these annotations can be queried. But Platypus does not directly entice the user to annotate pages: first, there is no direct added benefit from annotating (because one needs to make a separate query to see the results of annotating), and secondly annotating requires much user action: it is a separate activity, different from the normal writing.

Shawn Shawn [1] can be seen as semantic Wiki that addresses Platypus' shortcomings in both writing semantic annotations and in rewarding users for these annotations.

In Shawn authors write the normal text and the semantic annotations at the same time, in the same input field. Semantic annotations are syntactically distinguished from normal text by having the following form: `predicate:object`. All such statements are converted to RDF triples using the page on which they appear as subject. One can for instance have a page called `Shakespeare`, containing some text about the life of Shakespeare and his work. One line in the text may read `authorOf: Hamlet`, which will be interpreted as the triple `(Shakespeare authorOf Hamlet)`. This style of semantic authoring poses very little burden on the writer; users are free to embed semantic annotations at will and can do so while writing the normal text.

Annotating pages leads to instant gratification for users in two ways: navigational links are generated from the annotations and all semantic annotations can be queried.

Shawn generates sidebar links (both forwards and backwards) from the semantic annotations. For example, using the triple about Shakespeare and Hamlet, the page `Shakespeare` would show a sidebar link to `Hamlet (authorOf)`, and the page `Hamlet` would show a sidebar link to `Shakespeare (authorOf)`. If Hamlet was annotated with `rdfs:type Book` then the page would also show links to other books in the system. Users thus get rewarded for adding annotations: pages contain more information than explicitly written.

Shawn also allows users to query for pages containing certain statements using `predicate=object`; one can omit the predicate retrieving all pages that have some predicate with the specified object, and one can specify conjunctive queries by concatenating these statements. For example, `authorOf=Hamlet` would retrieve the page `Shakespeare`. Queries can be embedded in page text, prompting a live query each time the page is viewed. Users can thus create a persistent view of the data.

3.2 Characteristics

To summarise, semantic Wikis have Wiki syntax for page authoring and use RDF for annotation of pages. Users can annotate subtype relationships of pages and predicate:object statements having the page as subject. Based on the semantic annotation intelligent navigation can be offered, such as dynamic sidebar links to pages related via some predicate. Querying of the annotated data is possible through a (simple or powerful) query language.

4 Personal Wikis

An orthogonal extension of the original Wiki idea is the concept of a *personal Wiki*. Personal Wikis focus on providing extreme usability for personal information management. They are desktop applications and do in general not offer collaboration functionality. They serve as very lightweight note-taking programs, allowing users to related notes to each other by Wiki links. Although at first hand it seems self-contradictory with the collaborative nature of a Wiki, several of these applications have quickly become quite popular.

4.1 Examples

Tomboy Tomboy[3] is an open-source note-taking Wiki for the Gnome desktop (Linux) and very popular in this community. It is a very simple application: each note can be linked to other notes, or to email addresses and URLs. As in a normal Wiki, these links can be traversed; additionally there is a menu of recently accessed notes. Tomboy offers a full-text search to find relevant notes.

Tomboy is globally available in the desktop: it sits on the menu bar and one can open it with a single keystroke. Notes are automatically saved on each change, and there is an full undo stack for each note. These usability features explain its popularity: one can very quickly take notes in it, by just pressing the global key, typing some text, and pressing Escape to close it. There is no need for opening an application, opening a note, saving it, and rendering the result; all that happens on-the-fly.

Newton Newton[4] is another open-source personal Wiki for the Gnome desktop. It supports richer Wiki syntax than Tomboy, allowing for example numbered and bulleted lists. On the other hand, it is slower in usage: there is a editing mode and a viewing mode. In editing mode one sees the Wiki syntax, in viewing mode the page is rendered.

[3] http://www.beatniksoftware.com/tomboy/
[4] http://newton.sf.net/

WikidPad WikidPad[5] is a personal Wiki for Windows. It has a two-pane layout, showing a tree of all pages on the left hand side, and the current page on the right hand side. It supports a simple Wiki syntax for links and simple page markup, it has an auto-complete feature for Wikilinks, and supports full-text search. Users can assign categories to pages (to organise them), and annotate pages with arbitrary attributes (such as priority, or status of an item); these attributes can be used for aggregate views (e.g. all pages that contain todo items) or for changing system behaviour (e.g. export only pages marked as public).

VoodooPad VoodooPad[6] is a commercial application for MacOSX. It shows recently changed notes, backlinks from the current note to all that mention it, full-text search, auto-complete for Wikilinks. Additionally one can embed simple sketches into notes, and assigning categories to notes (to retrieve them more easily). VoodooPad also integrates with the MacOSX address-book: actions such as sending emails can be performed on recognised email addresses.

4.2 Characteristics

We summarise the characteristics of these various systems. Personal Wikis focus on quick note-taking like scenarios; they commit to ease-of-use by being simple and lightweight, and offering unlimited undo and real autosave. They are simple and have a limited functionality: users can type text, link notes, and perform full-text searches. Personal Wikis are tightly integrated in the desktop (global keybinding, system tray, application integration) and are thus desktop-specific.

5 Functionality analysis

We compile a list of possible functionality based on this analysis of existing applications and ideas about Wikis and their extensions. This list will serve as a requirements analysis for our semantic personal Wiki.

We can distinguish three main groups of functionality: functionality of authoring, retrieval, and navigation.

Authoring describes the available functionality to write, edit, and remove information; we distinguish Wiki syntax (for text layout), instant save (annihilating the need for explicit saving), full undo, global keybindings (making the application instantly available throughout the desktop), and application integration (allowing to reuse data available in other applications).

Retrieval describes the available functionality to retrieve available information; we distinguish full-text search, simple, powerful, and embedded queries, and logical inferencing (that returns implicit information).

[5] http://www.jhorman.org/WikidPad/
[6] http://flyingmeat.com/voodoopad

Navigation describes the available functionality to navigate through the system; we distinguish Wikilinks, intelligent navigation (using implicit information), keyboard navigation (enabling fast navigation), and instant update of the navigational entities (annihilating the need for explicit saving or refreshing).

Table 1 list these requirements in the above order, and indicate the level of support in the discussed systems.

	Wiki syntax	instant save	full undo	global keys	app. integration
Wikis	x				
Platypus	x				
Shawn	x				
Tomboy	x	x	x	x	
Newton	x				
WikidPad	x				
VoodooPad	x				x

(a) Authoring.

	full-text	simple query	power query	embedded query	inferencing
Wikis	x				
Platypus	x	x	x		x
Shawn	x	x	x	x	
Tomboy	x				
Newton	x				
WikidPad	x				
VoodooPad	x				

(b) Retrieval

	Wiki links	intelligent navigation	key navigation	instant updates
Wikis	x			
Platypus	x			
Shawn	x	x		
Tomboy	x		x	x
Newton	x			
WikidPad	x			
VoodooPad	x			x

(c) Navigation

Table 1: Capabilities of Wiki systems

6 Semantic Personal Wikis

Both extensions of the original Wiki concept offer distinct advantages. Semantic Wikis offer a better retrieval rate through intelligent navigation and querying,

but have mediocre usability. They are slow to start, since one needs to start a browser, surf to the Wiki, and wait for the server. They are slow in usage, since one needs to explicitly save and commit all changes, and wait for the required server round-trip. And they cannot offer a high-level of desktop and application integration since they are user platform independent; this enlarges the group of possible users but decreases the user experience. Personal Wikis address all these usability issues, and are specifically built to offer a very good user experience; on the other hand, they do not offer collaboration functionality.

A semantic personal Wiki is a combination between the best aspects of semantic Wikis and personal Wikis. It offers the usability of a personal Wiki, allowing users to quickly and unobtrusively write and semantically annotate pages. The semantic annotations are intermixed within the normal text, alleviating the need for a conscious change of mode. The Wiki entices users to add annotations by providing direct rewards in terms of intelligent navigation.

Such a semantic personal Wiki can be seen as a personal information system or as a lightweight ontology editor. It is not aimed towards collaborative work, but on providing usability and gratification: attracting people to use it and add semantics to their information.

6.1 SemperWiki

SemperWiki[7] is our implementation of a semantic personal Wiki; it is an open-source application developed for the Gnome desktop. We will give an overview of the system and explain its features in detail.

Overview The user interface of SemperWiki is shown in figure 1. On the left hand side the user can edit pages, on the right hand side the user can navigate; there is no separation between authoring mode and navigation mode. A page can consist of normal text, links to other pages or websites, and semantic annotations.

On the right hand side of the figure we see the navigation bar. This bar contains various ways to navigate through the system. First we see a "find" section, that allows users to query the system for pages containing certain statements. Below the "find" section links to various relevant pages are displayed. In this screenshot it only show the section "All pages", containing all pages currently in the system, other screenshots show more links as more relevant information is found by the system. Finally we see a history navigation section, that allows users to go back and forth in their navigation history; this feature is very useful if we "jump" into a link to edit some information, and then want to jump back again to where we were before.

Each page represents one resource and annotations state a property about that resource. SemperWiki stores some semantic information about the resources that are described on its pages, using the ontology shown in figure 2. Each resource is represented as a page, each page can contain some text and can have several outgoing links. Other information about a resource is added by the user (the standard graph is enlarged with the semantic annotations).

[7] see http://semperwiki.org/.

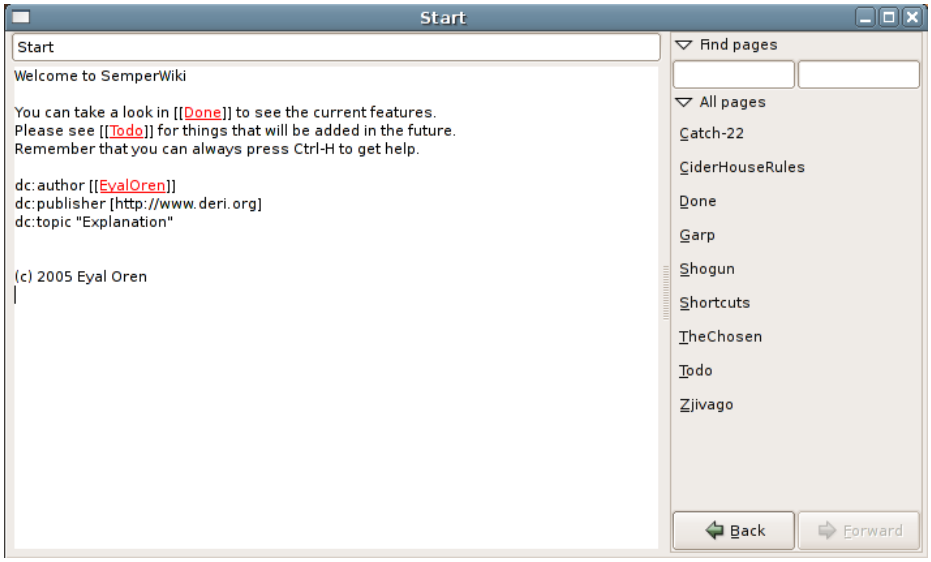


Fig. 1: SemperWiki user interface

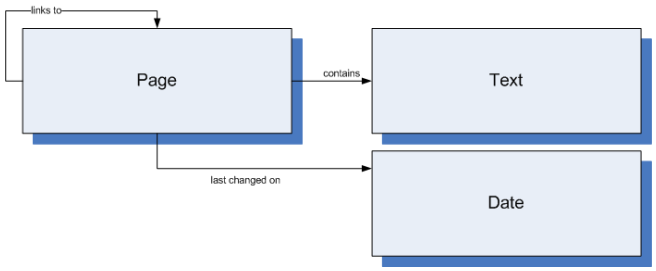


Fig. 2: SemperWiki ontology

SemperWiki stores all information in RDF[8]; we do this for two reasons: RDF is a very flexible representation and allows us to store various information about resources, and secondly RDF statements form the building blocks of the Semantic Web. The collection of triples that SemperWiki stores form a valid RDF model and can directly be exchanged with others.

In the RDF model, each page is identified by a URI, which is formed by prefixing its title with the base URI of the Wiki (defined by the user). Figure 3 shows the RDF triples that are stored about the page `Start`; for readability we have left out several triples.

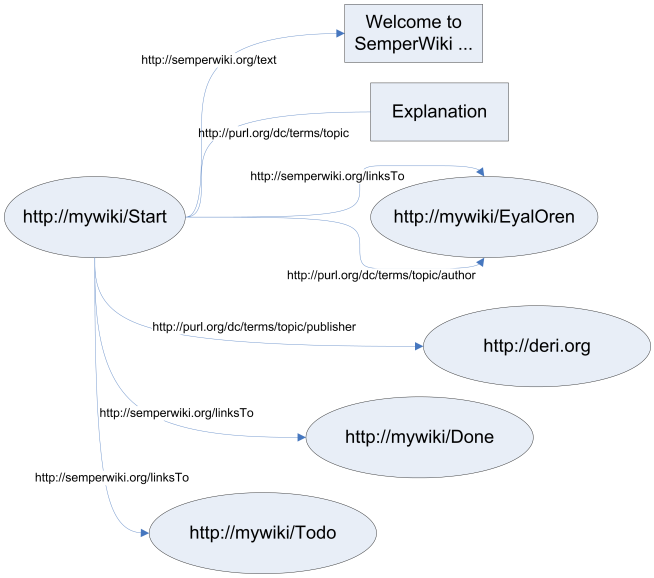


Fig. 3: RDF statements about the page "Start"

Syntax Figure 1 contains ordinary text, including links to other pages in this Wiki. We also see some semantic markup, stating that the author of this page is a resource in this Wiki (namely the page `EyalOren`), that the publisher is DERI, and that the topic is "Explanation".

For the semantic annotations we use a simple syntax. A statement is written on a line by itself and consists of a predicate followed by an object. Such a statement is expanded to a triple using the URI of the page as a subject. Predicates are resources, objects are resources or literals. Resources can be written as their full URIs, with prefix notation, or as Wikilinks. Wikilinks are expanded using

[8] http://www.w3.org/RDF/

the Wiki base URI, prefixed URIs are expanded using namespace abbreviations; these abbreviations are configurable through a preference dialogue.

The syntax (for respectively a full URI, a internal Wiki link, a prefix abbreviation, and a literal) is as follows: `[fullURI], [[WikiLinks]], prefix:localname, ‘‘literal’’`.

Navigation SemperWiki offers various ways to navigate to a page. First, one can click on any Wikilink to jump to a page. Second, if one knows the name of the page, pressing `Ctrl-G` will ask for the name and jump directly to it. Third, the history buttons can be used to go back and forth through visited pages. Fourth, the navigation sidebar will automatically show links to various related pages that can be visited by clicking on them; the next section explains how it works. And finally, one can query for any page containing statements by typing a predicate and/or object in the find section. An example query for all things written by John Irving is shown in figure 4, where the system found two relevant pages.

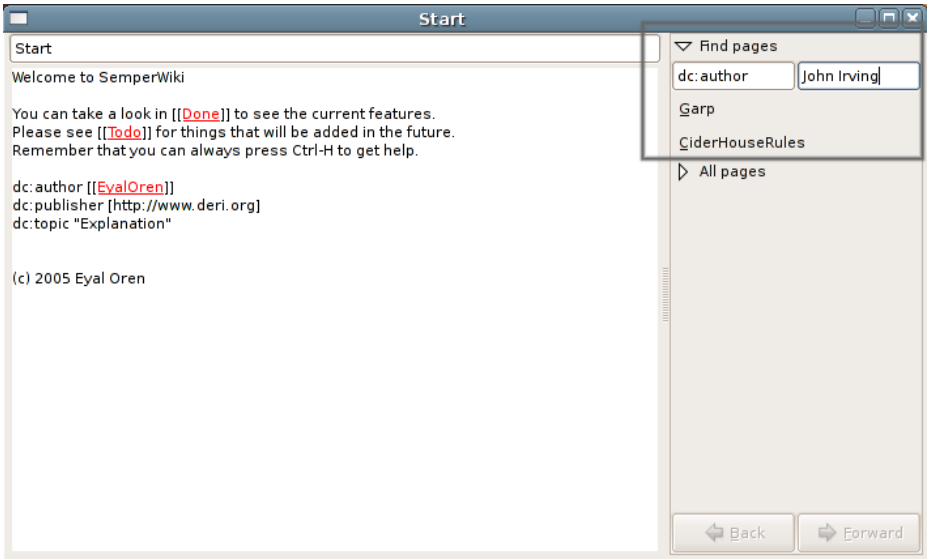


Fig. 4: Using the “find” section

SemperWiki is designed to help users navigate quickly and efficiently to their information. Therefore all actions in SemperWiki can be triggered by keyboard commands. These shortcuts can be accessed by typing `Ctrl-H`, as shown in figure 5. Also, as can be seen in figure 1, all links are assigned mnemonics (the underlined first letter) which triggers them with a keyboard command; for example `Alt-D` directly jump to the `Done` page.

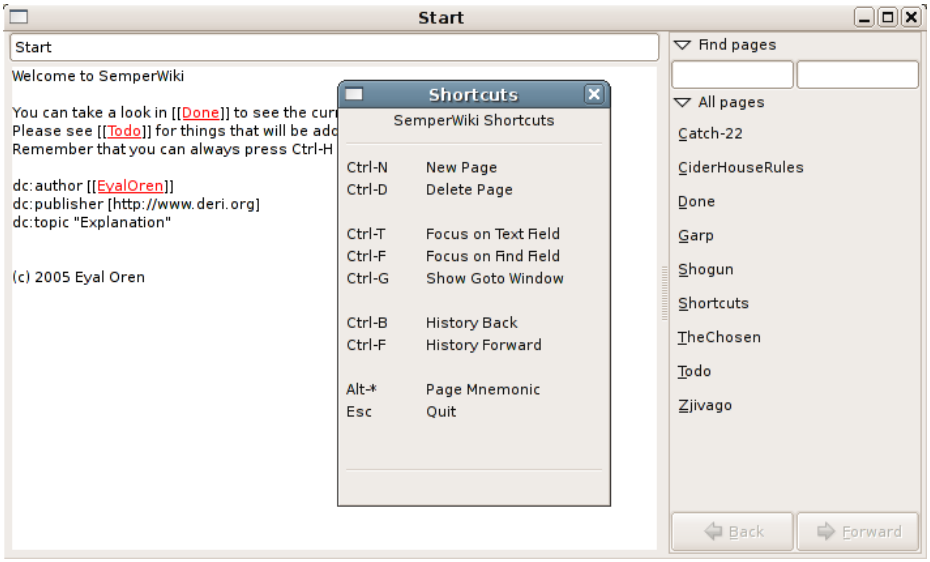


Fig. 5: Getting help

Dynamic Sidebar The links on the sidebar are generated automatically based on the available semantic information in the system. We show sets of pages that are related to the current one, and order these relations by the size of the sets of related pages. We show the most specific pages first (the smallest set of related pages), and gracefully decrease the amount of correspondence until we show all pages.

For example, the page `Garp` will show all books that also have John Irving as author (see figure 6). If there are any other predicates:objects in common we also show them, such as other books that are published by Random House.

Then we show for example all pages that are of the same `rdfs:type`; for Garp we show all other books in the system. This set is shown later, because it contains more results than those books that also have the same author as Garp.

We also display pages that have a reversed relation to the current one (but in a different colour). For example, the sidebar of the page `Book` would show links to all books.

Instant Response Everything in SemperWiki behaves instantly. All changes to all pages are saved instantly, without any user interaction. Likewise, the current page and location are saved instantly. Together this means the user can leave the system at any time (by pressing `Esc` and return the next time exactly where he left it.

All links are found and tagged while the user is typing them, without any need for saving or refreshing the page. All sidebar links are updated instantly upon any change in the page. This means that while a user is adding information to

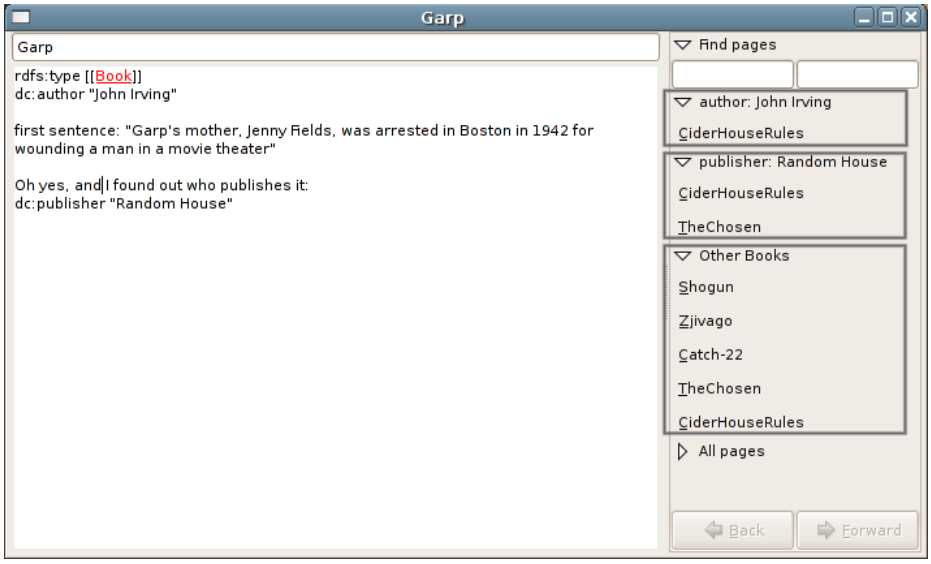


Fig. 6: Pages related to Garp

the page, the sidebar link is continuously up-to-date. If a user adds to some page the statement `dc:author "John Irving"`, the sidebar immediately updates to show other books by John Irving.

Having an instantly responding user interface greatly enhances the user experience. We can provide this responsiveness by caching all querying results and by using efficient regular expressions to parse the typed text. Currently, the system is very fast but we need to evaluate its scalability in terms of the size of the knowledge base.

Implementation The system is targeted at the Gnome desktop. As discussed, targeting a specific desktop is a trade-off between offering more users a lesser experience or offering less users a better experience, in which we have chosen the latter.

SemperWiki is implemented in Ruby[9], using the GTK[10] windowing toolkit for the graphical programming. It uses Redland[11] for RDF storage and retrieval. It currently consists of around 500 lines of code, many of which are related to programming the graphical widgets.

Redland is a mature RDF store, but it is still quite small so that it can be easily embedded in an application. It supports language bindings for many

[9] http://ruby-lang.org
[10] http://www.gtk.org/
[11] http://librdf.org/

different languages, including Ruby. We have also considered YARS [3], but at its current state it does not offer enough functionality.

The choice for the Gnome desktop was a personal one, but it is also the default desktop of Ubuntu, currently the fastest growing Linux distribution. The GTK toolkit is the default toolkit of the Gnome desktop, it is consistent and easy to use. Furthermore, it has excellent Ruby bindings.

The choice for Ruby was made for development speed. As an interpreted language it allows one to program very fast and develop prototypes very easily.

7 Evaluation

We can not evaluate SemperWiki against other semantic personal Wikis, as we are unaware of other systems. Also, we cannot do a user-based analysis of our tool, since the tool does not have an active user base yet. However, we can compare our current functionality against the functionality analysis in section 5.

It is clear that SemperWiki offers very little functionality in querying. That is actually on purpose: since we have no good solution for a simple but powerful query interface, we currently prefer the navigation sidebar: it is much easier to use. Furthermore, we can see that SemperWiki supports all encountered navigational capabilities, and almost all authoring capabilities; the missing "full undo" is actually planned for imminent implementation. Support for desktop application integration might an interesting discussion point for this workshop.

	Wiki syntax	instant save	full undo	global keys	app. integration
SemperWiki	x	x		x	

	full-text	simple query	power query	embedded query	inferencing
SemperWiki		x			

	Wiki links	intelligent navigation	key navigation	instant updates
SemperWiki	x	x	x	x

Table 2: Capabilities of SemperWiki

8 Conclusion

Managing the amount of information in our personal desktops is a challenge. The idea of a Semantic Desktop addresses this challenge by semantically annotating information chunks on our desktop, allowing better retrieval of relevant information. Since these semantic annotations have to be (partially) added by users, one must entice them to do so.

We have presented the idea of *semantic personal Wikis* to allow and encourage users to create semantic data. We have incorporated ideas from both semantic and personal Wikis: semantic Wikis focus on improved retrieval and navigation, personal Wikis focus on usability.

Our application SemperWiki allows users to manage their information: they create pages that contain both normal text and semantic annotations at the same time. They are instantly rewarded for these annotations by improved navigation possibilities. The system is simple, easy to use and responds instantly to all changes.

For future work we aim to address several issues. First, currently pages and resources have a one-to-one correspondence: each resource is represented by exactly one page. This makes annotations easier: statements are made using only a predicate and object, the subject is always the URI of the page. However, it makes editing complex data cumbersome since one needs to edit a new page for each resource, and it poses problems with blank nodes. Furthermore it prohibits making statements about external resources since all subjects have to be Wiki pages. We aim to decouple the relation between pages and resources, thereby allowing annotations of arbitrary resources (internal and external) and more complex statements.

Secondly, we want to improve the query and navigation interface, such that it is still easy to use and yet allows experienced users to make more powerful queries. For example, the list of related pages in the dynamic sidebar could be very long and we currently do not provide means to shorten or search through this list. For this we will look into related work such as the Longwell browser in the Simile project [6].

Finally, we aim to investigate possibilities to share the authored RDF data with others, either by publishing it on a persistent web location, or by sharing it over a peer-to-peer network with other users of SemperWiki. That way, what starts as a contradiction-in-terms for private use (a personal Wiki) returns to the Semantic Web.

Acknowledgements: this material is based upon works supported by the Science Foundation Ireland under Grant No. 02/CE1/I131. We thank Knud Möller and Stefan Decker for valuable discussions, and the anonymous referees for their suggestions of improvement.

References

[1] D. Aumueller. Semantic authoring and retrieval within a wiki. In *Proceedings of the European Semantic Web Conference (ESWC)*. 2005.

[2] S. Decker and M. Frank. The social semantic desktop. Tech. Rep. 2004-05-02, DERI, 2004.

[3] A. Harth and S. Decker. Optimized index structures for querying RDF from the web. In *Proceedings of the 3rd Latin American Web Congress*. 2005.

[4] B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley, 2001.

[5] L. Sauermann. *The Gnowsis – Using Semantic Web Technologies to build a Semantic Desktop.* Master's thesis, Vienna University of Technology, 2003.

[6] V. Sinha and D. Karger. Magnet: Supporting navigation in semistructured data environments. In *Proceedings of SIGMOD 2005.* 2005.

[7] R. Tazzoli, P. Castagna, and S. E. Campanini. Towards a semantic wiki wiki web. In *Proceedings of the International Semantic Web Conferenc (ISWC).* 2004.

End-User Application Development for the Semantic Web

Karun Bakshi and David R. Karger

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street
Cambridge, MA 02139 USA
kbakshi@mit.edu, karger@mit.edu

Abstract

Although a lot of information has become readily accessible and necessary for daily work, the current infrastructure for managing information is ill-suited for information-oriented activities: information and functionality are scattered across applications and websites, making it difficult to aggregate and reuse just the right set of content and operations required for unique user tasks. We discuss a collection of tools built into the Haystack platform that address many of the shortcomings of current applications, and allow composing reusable fragments of information and associated operations and views from the Semantic Web into a task workspace tailored to the user and the task. Users can change the workspace to immediately meet changing requirements to easily include, remove or reuse information in multiple tasks simultaneously. The time that a user invests for the initial setup and occasional updates to the workspace is amortized over the numerous times he or she returns to the task, and all relevant information resources are co-located and ready to use.

1 Introduction

Many information-based tasks require a person to assimilate and manipulate multiple pieces of information and the same information can be visualized and used in different ways in multiple tasks. Such tasks can range from a military commander monitoring troop movement, logistics support and weather feedback from a battlefront to decide on the next maneuver, to a doctor viewing a patient's past visit, prescription and x-ray history with respect to a diagnosis in order to determine how to proceed with a particular therapy.

Currently, we often tackle these tasks using applications developed to handle a range of closely related tasks. Each application offers users the opportunity to work with a certain pool of information by giving them a set of information views and a pool of operations that can be invoked on the information being viewed. But when an information management task does not exactly match one envisioned by the application developers, users find themselves fighting with the application, struggling to simultaneously display all the information they need, or to invoke operations that are buried deeply in the feature set of the application. Worse, users often find that the information they need is spread out over multiple applications. In such cases, users are forced to wade through cluttered desktops full of multiple application windows, each holding a small piece of the needed information (and lots of other distracting information).

The Semantic Web hints at a solution to some of these problems. The Resource Description Framework, its single, unified data model, is powerful enough to hold all of the information typically scattered across multiple applications. But merely unifying the data is insufficient; users must manage multiple sources of information in their tasks, and the amount that they can mentally manipulate is limited. To use the information to solve a particular task, users still need tools that will allow selecting and aggregating the information they need into a meaningful presentation (a task or information workspace) that lets them view and manipulate it as needed for their unique task. Such a capability might simply increase the efficiency of the task that would otherwise require juggling multiple applications (e.g., the doctor above). Or, it can be critical for success in a situation where multiple streams of information cannot be individually assimilated and their relationships deduced and acted upon under time pressure (e.g., the military commander). This capability becomes even more important for tasks that are long-lived or recurring such that the user must access the relevant pieces of information on multiple occasions.

In this paper, we argue that it is both desirable and possible to let end-users create their own information management applications (task workspaces) over the Semantic Web, choosing precisely which information objects they want to work with and how they want to view and manipulate those objects to complete a task. Such "end-user application development" would let them create workspaces designed specifically to solve their particular information management problems. We discuss a collection of tools built into the Haystack platform that address many of the shortcomings of applications mentioned earlier and allow composing fragments of information from the Semantic Web and operations that manipulate them into such a task workspace tailored to the user and task. We define a task workspace as a collection of information relevant to the task at hand that can be selected, presented and operated upon based on the user and task constraints, and that the user can aggregate in a lightweight manner. Our approach combines three elements:

- A **workspace designer** that lets users lay out the sets of information objects they want to work with in their application, specify which view to use to present each type of object and stipulate the relevant operations that should be readily available;
- A **view designer** that lets users specify how each type of information object in their workspace should be shown – what properties of those objects they want to see, how users should interact with them, and how they should be laid out; and
- A **channel manager** that lets users specify queries that dynamically maintain collections of related information that can be used to specify the relevant sets of information for one or more task workspaces.

Rather than specifying views, workspaces, and channels programmatically, users put them together using natural visual operations that they are already familiar with as tools for managing their desktop environments (such as clicking, dragging, dropping, and resizing). The workspaces, views, and channels designed by these end-users are themselves represented using RDF in the Semantic Web, creating an opportunity for users to share them with others, and for unsophisticated users to craft their "applications" by tweaking preexisting ones instead of creating them from scratch.

We have implemented our system as part of the Haystack information management platform. Haystack provides a set of cooperating technologies and tools supporting end-user creation, visualization and manipulation of Semantic Web content, as well as application development for the Semantic Web [23, 24, 25, 28, 29]. Along with a blackboard-style RDF store, it hosts agents to provide automated reasoning and supports a user interface framework that provides pervasive context menus, drag-and-drop capability, and a view architecture that appropriately selects and presents views for different types of information entities depending on the context of use.

1.1 Motivation

Although a lot of information has become readily accessible and necessary for daily work, the current infrastructure for managing information is ill-suited for information-oriented activities: information and functionality is scattered across applications and websites, making it difficult to aggregate and reuse just the right set of content and operations required for unique user tasks.

Consider a software project manager who needs to manage the tasks her team does, as well as the budget and schedule for the project. She might need to manage contact information for team members, to-do list and assignment of action-items to individuals, e-mails corresponding to the project, outstanding bug reports, a budget spreadsheet and a schedule. In order to react to a customer e-mail about a software bug, she must switch from the e-mail client to the bug tracking software to enter a bug report. Then, she must switch back to the e-mail client, recall which software developer would be best suited to fixing the bug, look up his/her contact information, and send him/her a bug report number. The project manager may also need to meet with the developer, and hence negotiate a meeting time via e-mail, and update her personal calendar. After the meeting, the project manager will need to switch to the scheduling application to update the project schedule to reflect the time that will be consumed in fixing the bug. Another application switch may be needed to update the developer's outstanding tasks if the scheduling software does not support to-do lists for team members. Finally, she must switch to the spreadsheet software to update the project budget to take into account the resources consumed by the bug fix. As this example demonstrates, rigid applications restrict users to behavior patterns that force them to become passive recipients of information

rather than being able to actively mold it to impose their own world view. They are forced to bridge the gaps between applications by having to:

- manually collect information by opening various applications;
- reenter the data elsewhere (e.g., retrieving information from one application using an identifying key from another or duplicate it in order to manipulate it in another application), acting as the glue between applications because applications do not understand each other's native information format;
- mentally select and associate items of interest, and reason with them, since they cannot be easily juxtaposed; and
- dig deeply into menus and multiple dialog boxes to find just the right operation to invoke.

Although, a single application could be developed that included functionality for all project management tasks, it would still be a rigid solution. What if the project manager also wanted to use the new application's functionality, such as calendar and e-mail, outside of the project management application, e.g., to see when her dentist's appointment is, or to receive mail from her spouse? What if she now had to also track news about a competitor's products? What if she wanted to define a new "task" that involved approving and disapproving bug fix recommendations that aggregated *just* the bug information from the various project workspaces into a new workspace? All such continual user requests for customizations would be difficult to satisfy by a limited number of application developers.

Today, the World Wide Web is an indispensable information resource and the Semantic Web, with metadata annotated information, will be even more vital for completing information-based tasks in the future [1]. On the Semantic Web, information will be produced faster using automated means, with finer access and semantic granularity and shared via web services than can be handled manually. Attempting to manage this torrent using multiple applications instead will lead to a proliferation of specialized applications that fragment information arbitrarily, making information management more complex, time-consuming and error-prone. A more robust solution is needed that does not require significant developer support to easily adapt to and reuse unanticipated types of information fragments for unforeseen tasks as soon as they become available on the Semantic Web.

1.2 Organization

The remainder of this paper is organized as follows: first, we provide a detailed system description. Next, we discuss related work in various areas that impact our current work, followed by a discussion of the high-level approach to developing a task management infrastructure. Then, we present important design and implementation decisions, followed by a summary of the salient benefits of our tools. We conclude with an outline of future research goals.

2 System Walkthrough

Consider Ann, a neurologist investigating brain structures. Figure 1 shows a task workspace that she has designed to assist in the task of writing a research paper attempting to identify the correlation between seemingly unrelated symptoms and mental illness diagnoses, based on the volumes of various brain structures. (The example is based on actual data from the Internet Brain Volume Database that describes the volumes of sundry brain structures in test groups having various mental illness diagnoses, and the research papers that report such data [31].) She has designed a workspace to help aggregate various information resources required to support the hypothesis and the paper writing process; she has defined a query of all Caudate instances having a volume of greater than 9 cc as potentially intriguing (top left). In addition, she has defined a query for the groups to which this data correspond, by having the latter query (bottom left) extract the group related to each member in the first one. Also, Ann is interested in seeing the publications that reported the relevant data (top center), the other publications by authors of the reporting publications (bottom right), as well as a list of collaborators for the paper writing task (top right). Ann chooses to view the Caudate and Group data using a small view that exposes only certain relevant

properties so that she can easily glance at multiple items and quickly identify interesting ones. The publication view shows the title of the paper, but also allows her to click on any of the authors to get additional information about him or her. The view used for people who are authors of the interesting publication reuses the publication view to show a collection of their publications, which is the only aspect she is interested in knowing about each author in her current mind-set. On the other hand, the view she uses for people who are her collaborators simply shows their name as she is in frequent contact with them and does not need additional information yet. As can be seen, frequently used operations relevant to the information in the various winlets are within easy reach. (We coin the term winlet to refer to a tile in the workspace that captures particular content and relevant operations; e.g., the Interesting Publications winlet.) Clicking on the operations implicitly uses the currently selected item in the winlet as a parameter if it matches one of the required parameters for the operation.

Fig. 1. Initial Paper Writing Workspace in Usage Mode

As she comes to work this morning, Ann decides that the past publication history of her collaborators would be useful to have close at hand. In addition, since the paper sub-tasks are getting to be numerous, she decides that it's time to start tracking them via a to-do list. Also, since her spouse has taken their son for a doctor's appointment, she wants to be kept apprised of e-mails from home. In order to handle these changes in her task description, she switches the task workspace to design mode (Figure 2) using the "Change View" menu, and adds a preview window to link with the Collaborators winlet in order to show the papers published by the selected collaborator (far right, below Collaborators winlet). The various layout manipulation operations (split horizontally, split vertically and delete) available on the toolbar to the right of each winlet can be used to lay out the workspace as desired. Clicking on the maximize button expands a winlet to take up all available space when configuring its details. Each winlet can be named and resized. In addition, winlets allow one to configure the content (Content Pane), change how it is viewed (Presentation Pane), and how it can be manipulated (Manipulation Pane). The Content Pane allows the user to specify which particular information object(s) are to be shown in a particular winlet. They can be specified either explicitly by identifying a particular entity, or implicitly by specifying a query (implying a collection of information objects) or the currently selected item in another winlet. The Presentation Pane allows the user

to select the particular view to be used to render the information object(s). (Each type of object can have multiple associated views, e.g., a small summary view versus a larger, more detailed one.) Finally, the user can drag operations that are frequently used and/or applicable to the information objects being shown into a collection in the Manipulation Pane.

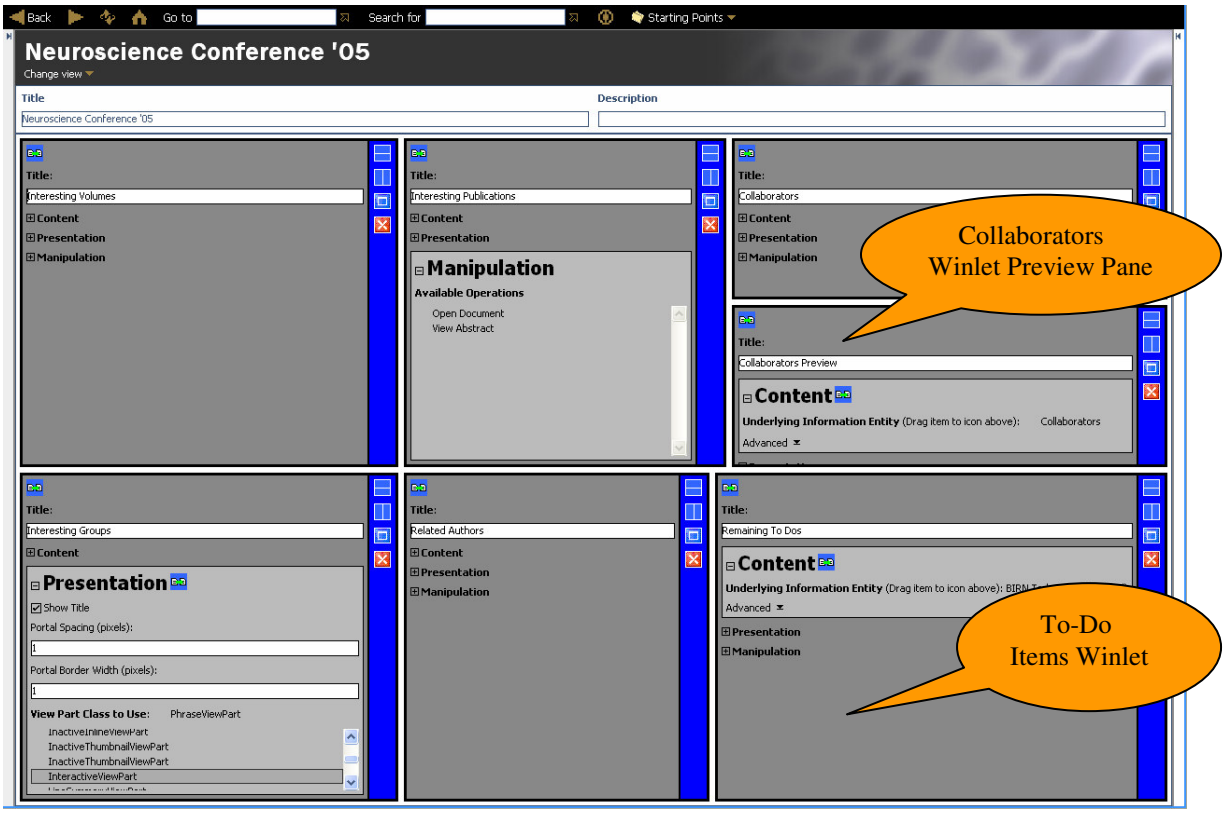


Fig. 2. Paper Writing Workspace in Design Mode

Ann then uses the channel manager (Figure 3) to define a new information channel (query) for to-do items for the paper, and adds a pane in the task workspace to track this collection (bottom right of Figure 2). A query can be defined by dragging and dropping a query primitive onto the channel. A query instance is instantiated and the user may then specify values for its parameters. In order to simplify the query building process and reduce usability barriers, the channel manager allows one to copy existing channels and interactive evaluation of the query by allowing the user to tweak query parameters as necessary and observe the updated results. Finally, in Figure 4, Ann changes her query for interesting volumes to look at Amygdala volumes (channel viewer docked in right pane), changes to a more detailed view for the groups reporting the data (bottom left), and docks an information channel viewer that only shows e-mails from home outside the task workspace (right pane), but within her field of view. The left pane in Figure 4 shows the final task workspace in Haystack. Note that users need not commit all information they wish to work with (even if temporarily) into the workspace if it is not relevant to the task. (The right pane in Haystack can be used as a scratchpad to temporarily place any entity within view.). The notion of channels allows selective interruption to be made possible as Ann has allowed herself to be temporarily informed about e-mails from home by docking a channel viewer tool in the right pane.

If a query's parameters are changed (e.g., she decides to investigate the Amygdala brain structure rather than the Caudate), all dependent queries will also be recomputed, causing a ripple effect in the workspace as it updates periodically. The reader will notice that the content of the various winlets relying on dynamic channels have changed appropriately in Figure 4. Thus, the workspace stays current with respect to the current query definitions as well as information corpus, enforcing the relationships between the various winlets and simplifying data exploration – a task that would otherwise require significant manual re-querying and correlating if the corpus or the query had changed. In addition, Ann has decided to look at

more detailed information about the new groups she will now be perusing, and hence has changed the view that should be used. She has also chosen to use the author view for her collaborators in the preview pane. Note that although no software was developed for the "paper writing application," Ann was able to create it with relative ease. Furthermore, the "application" is amenable to change as user needs change.

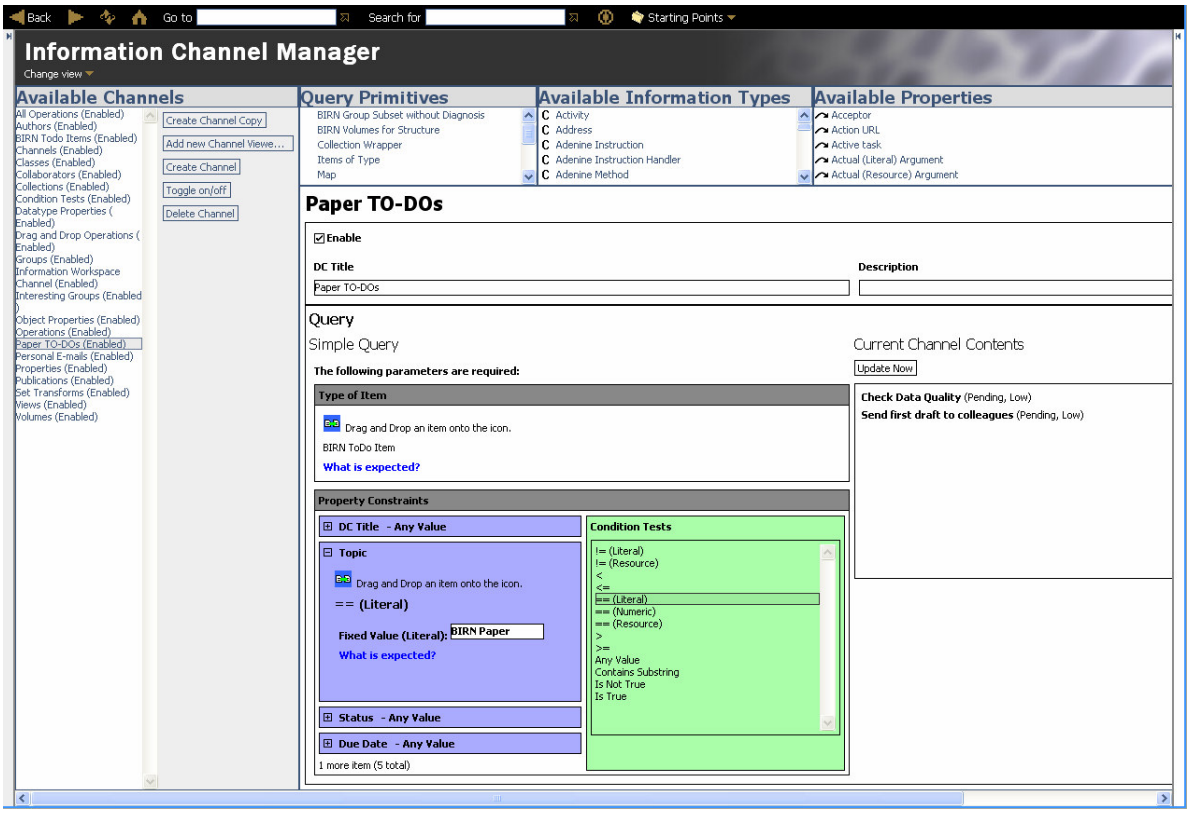


Fig. 3. Information Channel Manager

3 Related Work

In this section, we attempt to highlight some important milestones that have marked two orthogonal trends that collectively have a significant bearing on our work: the evolution of information management workspaces from collections of application windows managed by window managers to applications with task focused interfaces, and the increasing sophistication and ability of end-users to move from merely configuring their information management tools to directly specifying the content, presentation and manipulation aspects of their information management tasks.

That users would require multiple information resources to complete a task became clear with the advent of the earliest windowing systems and toolkits used to build applications that could be simultaneously active on the desktop. In addition, a realization that users have different needs and preferences in how their available UI real estate needs to be allocated to these various resources led to the development of window managers, each providing users with assorted tiling and resizing semantics [2] (see [7] for a survey). In addition to window managers that allow flexible control over UI real estate usage, much work has also been done in allowing users to allocate their available space as efficiently as possible [3, 4]. QuickSpace, for example, implements simple window management operations to allow users to quickly allocate greater space to their primary operating window while maintaining the overall layout of the desktop.

Additional experience with ephemeral window managers that required repeated work to set up a task workspace led to more persistent options such as virtual desktops, which realized the value of capturing user task context in a returnable environment. The windowing systems for the Microsoft Windows and Apple Macintosh operating systems comprise two such commercial efforts that can provide one desktop

per user by default. Some flavors of Linux extend this feature set by allowing multiple virtual desktops per user that are persisted even across system reboots. A more involved research effort, Rooms, extended the concept of a virtual desktop further by allowing users to create a separate user task workspace (a room) that specifies which tools are open, as well as the layout and presentation of their windows [5]. These rooms (which could be shared between users) allowed capturing settings implicitly (e.g., window locations or sizes) as well as explicitly (e.g. connections between rooms). Rooms could share control panels of common tools and information. However, Rooms and other window management functionality operated at a coarse information management granularity; they allowed managing windows of entire applications as opposed to just the subset of an application's information and operations that were relevant to a particular task.

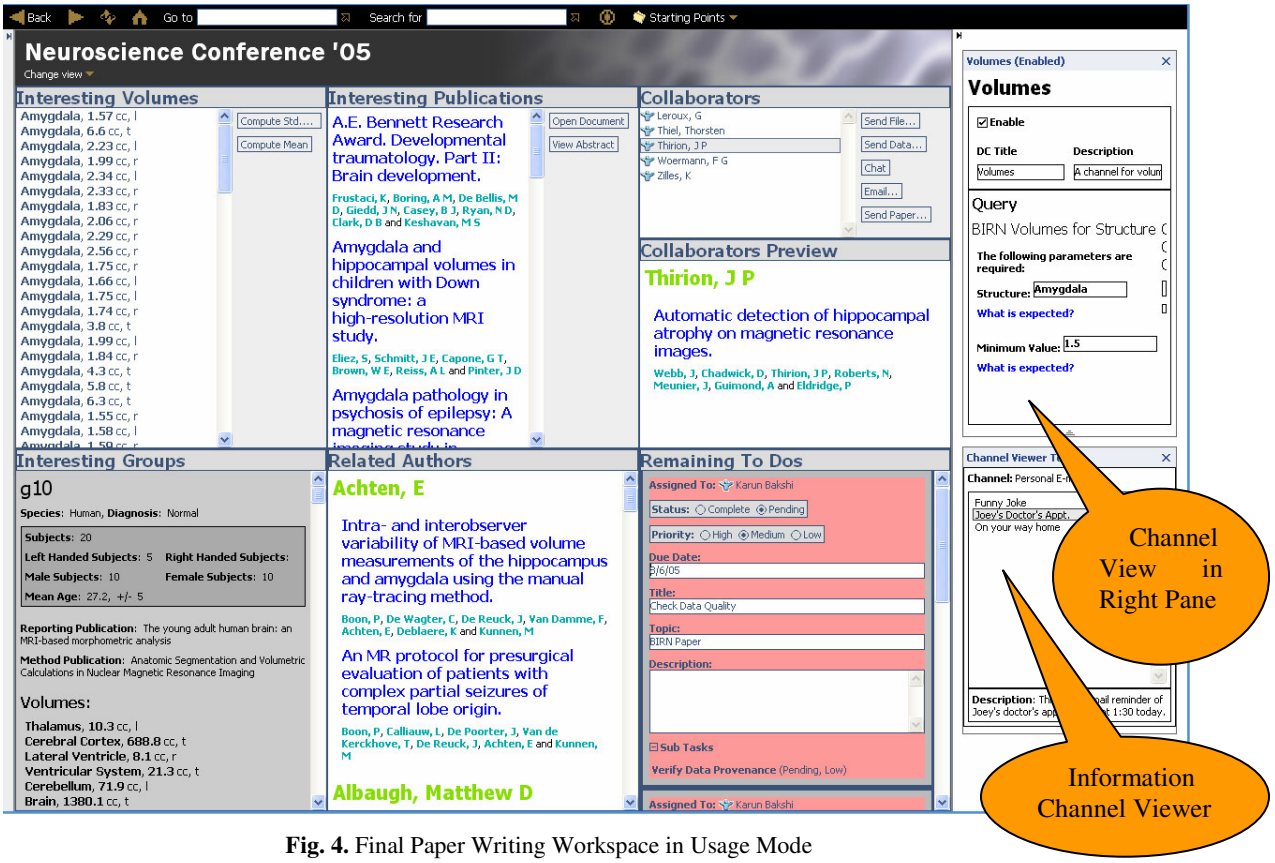


Fig. 4. Final Paper Writing Workspace in Usage Mode

As user tasks employing multiple applications or different parts of a single application have been elucidated, application user interfaces have progressed from merely allowing information manipulation to interfaces that are specifically designed to assist in a particular task; many window management ideas have seeped into single applications for tasks that used to require multiple applications, yielding interfaces with heterogeneous but focused subsets of information and operations juxtaposed in a single presentation and specifically tailored to a particular task. Taskmaster is a Visual Basic add-on for the Microsoft Outlook client that targets project management activities in mail clients [8]. It makes information resources easy to access "at a glance, rather than scrolling around inspecting folders" by taking advantage of the heuristic that items in the same e-mail thread generally correspond to the same task. Thus, incoming, outgoing and draft messages are grouped into a project context based on such message data. Although the ideas embodied by Taskmaster do increase usability of information from the perspective of content customization, the benefits are restricted to a predetermined domain and users have little control over the layout or other presentation or manipulation capabilities. Also, users have little queryable control over the content; the developers have already decided on the content queries whose results will populate the project interfaces. The Kubi Client is a commercial effort in the same vein [9].

Task workspaces can be created and maintained by users, in addition to developers. Web Montage employs a subtle, implicit approach to creating task interfaces; it observes users and uses machine learning

to build a personalized user web portal for web browsing [10]. However, the aggregated content is read-only, not manipulable. Similarly, the Microsoft Office suite of applications implicitly supports user tasks such as reconfiguring the user interface to only show recently used menu items by default, presumably employing the "most recently used" heuristic as a means to approximate the required task resources [11]. However, learned interfaces are error prone, and take time to adapt. Also, such interfaces do not respond well to interleaved tasks that force them to unlearn the settings from the previous task.

User sophistication has also increased over the years; applications that only allowed configuring of simple rendering preferences have ceded greater explicit control over the task interface creation process to the increasingly experienced average user. E-mail clients such as Microsoft Outlook, for example, support one aspect of task definition – content – by allowing user-defined rules that trigger based on message arrival or message sending events to maintain dynamic collections [11]. Hunter Gatherer is a recent system developed at the University of Toronto that simplifies capturing parts of an arbitrary web page (again, read-only content) into a contextualized collection [13]. Similarly, Presto, an interactive document management system, allowed defining tasks interfaces based on sets of relevant individual documents and other relevant, document collections, where each document can appear in multiple workspaces [21]. The document collections themselves are dynamic, and described via a query that utilizes underlying key/value attribute pairs that facilitate organization, search and retrieval of documents in the corpus. Although they support greater user influence, these examples are limited by the range of task workspace aspects that can be controlled, i.e. they allow controlling just the content.

Other examples that allow control over the content, layout and manipulation capabilities exist, but are limited by the task domain. For example, content portal (e.g., *MyYahoo!*) and news organization websites nowadays provide a rich set of primitives that can be used to create personalized webpages which allow users to filter and/or aggregate the content provided by the underlying organization(s) as well as specify its presentation for the news reading task [12]. Relevant operations such as search or e-mail are also close at hand. Microsoft Visual Studio, on the other hand, allows the layout of various fixed content panes to be controlled [30].

Generic workspace building tools for arbitrary domains also exist. With the advent of the World Wide Web and other queryable multi-media repositories that require direct user access, being able to shape the result set of information via a good query/result interface has steadily gained more attention. Delaunay^{MM}, a querying framework for distributed, heterogeneous, multi-media data stores allows users to specify a query and layout preferences for its results without *a priori* knowledge of the underlying schema or query language [14]. Similarly, Snap-Together Visualization (STV) allows users to query a relational database and link the result set visualizations in support of a complex information exploration task [15]. However, the STV system required users to understand database concepts and directly manipulate the MS-Access database and associated queries, making it less usable by lay people. Both Delaunay^{MM} and the Snap-Together Visualization system represent simple user workspaces as users can specify content of interest, along with presentation preferences. Nevertheless, although both examples can be employed in arbitrary domains, they are constrained by a particular database instance (or small sets thereof) and its associated schema(s). That is, they cannot scale easily to capture multiple domains with heterogeneous or semi-structured data models.

Recently, the notions of task workspaces and user control over their specification have merged in the WinCuts project that allows users to aggregate only the relevant portions of source windows into a new task context [16]. Each WinCut is a live connection to the source window and can redirect user interactions to it. Thus it allows visualization and interaction with information fragments as in the original application, but it does not provide direct access to the underlying semantic entity. Nevertheless, it advocates and supports the need for users to work with subsets of information found in arbitrary applications.

With the Semantic Web in its formative stages, much attention has been paid to (ontology and content) authoring on the Semantic Web as was paid in the original World Wide Web [17, 19, 20]. Recently however, more attention is being devoted to using semantics to enhance the end-user browsing and navigational experience for specific domains [18, 22]. General UIs for Semantic Web browsing have also evolved from simple RDF graph viewers [26] to semi-structured data browsers [27]. With the content on the Semantic Web anticipated to be annotated at a much finer granularity than documents (as in Presto) and the ability to capture and relate arbitrary domains, we foresee an opportunity to merge task workspaces, user control by leveraging semantics to allow creating task workspaces that allow semantic rather than pixel-level interaction. It is this vision for information management on the Semantic Web that we hope to

realize with our system. In the process, we borrow from many of the ideas discussed above and learn from their limitations to inform our approach and implementation.

4 Approach

The software project manager's example starkly illuminates the gap between how users think about the task and supporting information (either because of personal preferences, task needs, etc.) and how supporting applications are designed based on *a priori* developer assumptions for a limited range of user, task and domain scenarios that rigidly define the boundaries for what can be accomplished. The design and implementation of these applications fail to take into account the fluid nature of information management tasks; the required information, visualization and operations change as different users define a particular task differently, tasks definitions change over time (e.g., to include information from other domains), varying levels of expertise require either more or less information to complete the task, and users may have different preferences for visualization and interaction. Yet, the supporting tools are static in nature, rigidly bottling up information in application containers. Information producers have most of the control over how information is packaged, presented and manipulated. The information consumer (whose productivity these decisions significantly affect) has little say in these decisions.

We posit that giving users control over three primary aspects of a task definition will grant them sufficient power to eliminate many of the shortcomings of the current information management infrastructure. Users need to be able to define and modify the content, presentation and manipulation capabilities of their task workspaces.

The information content that users require, devoid of its presentation, is perhaps the most critical ingredient of the task that the user is attempting to accomplish. Yet, users often lack the ability to easily select the subset of the information corpus to show and manipulate in a given context. On the Semantic Web, this issue will have even greater import as the authoring ontology is not intended to necessarily match all usage scenarios, and hence users must easily be able to extract the relevant portions of the information efficiently; the authoring ontology should not restrict the applicability of the information.

User control over information presentation is also crucial, and a significant portion of the functionality in many applications today is devoted to allowing users to modify the presentation of the information in a manner consistent with how they want to manipulate it. We consider the problem of information presentation as consisting of two parts: specification of high level layout of multiple types of information and specification of the view used to display a particular information entity. The layout capability relates to the ability to aggregate and co-locate arbitrary information from various sources – a capability that would have allowed the software project manager in our earlier example to avoid "hopping" across applications as relevant information could have been aggregated into one interface. Users should also be able to select or create views of the underlying information that are relevant to the task at hand. Given the same information, not everyone manipulates it or visualizes it in the same way either. For example, a person in the accounting department may be interested in the spreadsheet view of some sales figures, whereas a higher level executive would prefer a chart based on this data.

Finally, many decisions about operations (how the information is to be manipulated) associated with the information are made by information producers: certain operations may only be available in certain views, or have certain means of access (e.g., toolbar, menu item, etc.), force re-entry of the same parameters each time they are invoked, etc., and users must thus adapt how they perform a task based on this static aspect of applications. The user, who actually determines the frequency of the operation invocation based on his/her task, has little say in these important decisions.

In order to provide users with the type of flexibility to create fluid task workspaces that can take advantage of the rich semantic and application-independent information on the Semantic Web as soon as it becomes available, what is needed are small units of user interfaces, and application logic that they can easily combine to yield larger, more powerful task interfaces. We have developed several tools that enable the creation and stitching together of such reusable application fragments in order to provide users with greater control over the three high level aspects of tasks discussed above.

Our tools are situated in Haystack, a general information management platform for the Semantic Web that provides many of the building blocks we require at the system level. We discuss the important design and implementation details of Haystack and of our tools below.

5 System Design and Implementation

In this section, we discuss some of the important design decisions we made and implementation techniques we employed in building the various tools that support end user workspace creation. The primary components include a workspace designer that allows aggregating and laying out custom content and operations into a workspace, an information channel manager that makes it possible to specify queries for dynamic collections of related information, and a simple view designer that allows users to specify how to present underlying Semantic Web entities. But first, we briefly discuss the desirable ideas embodied in the Haystack platform and the building blocks it provides for our tools.

5.1 Haystack Platform

Haystack encompasses and makes available several key ideas and components that support creating, visualizing and manipulating Semantic Web content. At its core, Haystack employs a single data model consisting of a semantic network expressed using the Resource Description Framework (RDF), the standard for knowledge representation on the Semantic Web. It provides a language (Adenine) for simplifying expression, manipulation and querying of RDF data. Furthermore, imperative Adenine code that manipulates the data can itself be compiled into RDF data using a target, portable, runtime ontology akin to Java bytecodes, thereby rendering a majority of the Haystack system declaratively specified. Adenine serves as the lingua franca of the system, enabling communication between (and implementation of) its various components via the generic blackboard-like RDF store. Haystack can also host services that automate various tasks, e.g., categorization, summarization, extraction, learning, recommendation.

Haystack provides a user interface framework that recursively renders views of an information entity by rendering the views of other information entities that comprise it. It consists of the Slide Ontology, an extensible, HTML-like ontology for content layout and rendering that is used to create views of information entities. An information entity is accessed by browsing to it (i.e., Haystack navigates to the underlying Uniform Resource Identifier on the Semantic Web). A view for an entity is automatically selected from the pool of applicable views by Haystack based on the entity's type, and the context of use (e.g., available UI real estate). Furthermore, the UI framework supports context sensitive manipulation such as context menus and drag-and-drop operations. Finally, imperative code in Adenine can be exposed to the user and invoked by him/her via operations. Operations are parameterized Adenine methods that perform a task for the user using the specified parameters. Relying on metadata annotations on the operations themselves, Haystack employs an automated technique (UI Continuations) for instantiating instances (closures) of the Adenine method calls and collecting the relevant parameters from users. Also, operations may be curried, i.e., the user may customize an operation by specifying values for some, but not all, parameters, but not all [6]. The resulting curried operation can be saved as a new operation, and used as a template for applying the operation in new contexts, that all share the same value for the saved parameter, but require the remaining parameters to be specified during each subsequent invocation.

5.2 Workspace Designer

The workspace designer in Haystack allows creating, copying, deleting and configuring instances of task workspaces. A task workspace in Haystack is data. It follows a certain ontology that captures the layout of the comprising winlets as well as their associated properties. Each winlet allows the user to specify the content to show, how to present it, and which operations should be co-located (corresponding to the three aspects of task spaces we sought to give users greater control over). A Boolean flag allows disambiguating how the underlying content is to be interpreted: as itself or the entity it represents, e.g., should the underlying query parameters be shown, or its results? The view specified by the user is then applied to the appropriate interpretation. Finally, operations that the user specifies can also be curried versions of other operations.

A task workspace has two primary views: a design view that lets users edit its properties, and a usage view which interprets its properties to render a task workspace. It is rendered by recursively rendering each of the winlets using the appropriate design and usage views. Users can iteratively prototype a workspace by

switching between its design and usage views. Finally, workspaces, like channels and views can be copied to avoid having to start a new one from scratch.

Several important design decisions were made in the design of the workspace functionality. Perhaps the most important design decision that affected the construction of workspaces by users was how to allow users to allocate space to winlets. We chose to facilitate a tiled layout engine rather than allowing free floating windows. Given our understanding of the portal creation task from *Yahoo!*, and the ideas espoused by the QuickSpace project, it seemed that the goal of user controlled content layout was to allow users access to a canvas that is appropriately segmented and completely devoted to showing the content of interest, with the user being able to specify the location of items, and possibly their size. Such an approach allowed users to allocate maximal space to the content of interest. Furthermore, it allowed the space to adapt to a local change: if a user increases the amount of space for one item, the space for other items automatically decreased without overlapping – an important feature desired by users, as demonstrated by QuickSpace. The decision to only allow the user to split cells into two rows or columns was driven by the desire to keep the operation of segmenting the usable space as simple and efficient ("one-click") as possible. Similarly, we chose to allow users to place operations in the right pane of a winlet rather than place them in the context menu in order to minimize user clicks when invoking them. Arguably, the tiling functionality could be made more sophisticated and using smaller icons for the operations would minimize the wasted space.

We chose to allow modality in workspace interaction (design vs. usage modes) primarily because the act of specifying underlying content for a winlet was most effectively done by using drag-and-drop, which required modality in order to be able to disambiguate between the act of specifying the underlying content and dragging an item onto the winlet in usage mode. Thus, we felt it would be easier for the user to have a clear separation of modes rather than be distracted by extraneous mode specific widgets which would also waste UI real estate.

5.3 Channel Manager

Whereas a particular entity can be specified to be shown in the workspace explicitly via a drag-and-drop operation, and Haystack supports creation of collections by the explicit specification of its members by the user, the channel manager allows an implicit collection specification mechanism by providing the user an interface to query the underlying RDF store. As in Presto, collections defined by channels are self-maintaining, organizing mechanisms for dynamic corpora of information that allow users to specify a query that maintains an up-to-date dynamic collection of items; they allow the user to impose his/her world view on an otherwise raw and changing set of information by defining a set of persistent indices of relevant information that are then maintained up-to-date by the system. Thus, given a store being modified by the user, agents and the arrival of new information, the user can create a stream of information that is important to him/her independent of the source or creation method of the information. Channels can then be treated as standalone units of content that can be used to specify the underlying content of a workspace winlet.

The Channel Manager allows creating, copying, deleting and configuring channel descriptions. A channel in Haystack, like task workspaces, is also data. It follows the channel ontology, which captures whether its title, description, whether it is enabled, its query, and the current query results. The channel shown in the channel manager earlier showed its editing view in the bottom right of Figure 3.

The Channel Manager functionality can be fundamentally divided into the processing and user interface components. The processing component can be further subdivided into the channel manager agent and the set of available query primitives. Channels are computed by a channel manager agent (running as a Haystack service) that periodically updates all channels by evaluating their queries. A query is simply an Adenine function with various parameters. The query primitives available to the user always return a set of URIs of matching entities. Any new Adenine method that meets the above constraints can be exposed as a query primitive to the user and immediately interoperate with the channel manager agent using appropriate annotations, thereby allowing the set of query primitives to be extended. Dragging a query primitive to a channel results in a new instantiation of that Adenine method call: a new query primitive closure. The channel then refers to this closure as its underlying query. The view for a channel consists of sub-views of the underlying query primitive instantiation that allows the user to modify its associated parameters.

Current RDF query primitives allow using the associated metadata annotations of the underlying RDF to "carve" the information by selecting out only the relevant portions. The simple conjunctive boolean query

primitive which allows the user to select entities that meet constraints on literal valued properties is perhaps the most common query primitive. Users can specify string matching, numeric relation, or boolean typed condition tests for the literal valued properties; items matching all the conditions (boolean conjunction) satisfy the query definition. The conditions tests, like the query primitives, are also special types of Adenine methods and can be similarly extended. Another useful primitive is the Map primitive that allows mapping a property over all members of a source channel to generate a new channel of entities related to entities in the first channel via the mapped property.

Along with other RDF query primitives that enable simple and common RDF querying use cases such as selecting items of a particular type, etc., a set of channel operator query primitives (e.g., set union) are also available that allow combining channels or user-specified, fixed collections to create new channels.

Finally, developers can add new primitive types that support complex domain specific queries or use domain specific terminology simply by writing the query code, and, if the default view is deemed too esoteric, designing a custom view to prompt for relevant parameters.

The decisions to disallow duplicate entries, ordering preferences, and projecting properties of the returned items were made in the hopes of limiting the complexity of decisions a lay user would have to make. Also, we felt that specifying a single view that shows a particular set of properties for an underlying entity would be easier for the user than specifying how to view the tuples of property values resulting from a projection of its underlying properties.

5.4 View Designer

When working with information, it is critical that users be able to customize views not just in terms of cosmetic properties (e.g., color and font) or pre-determined/fixed, domain-specific operations (e.g., sorted e-mails by sender) deemed to be useful by the developer, but also by controlling which set of properties are accessible and how they can be visualized and manipulated in conformance with the semantics of the underlying information entity.

We developed a domain-independent, baseline view designer that lets users inspect any properties of any information entity in Haystack, by taking advantage of minimal information semantics. The view designer allows users to create metadata lens views that select and render values of properties (metadata) of interest from the underlying entity. The metadata lens views share the same layout engine as in the workspace functionality and also have similar design and usage views. Thus, users can segment a rectangular area into winlets, and instead of content, presentation and manipulation setting, specify a property to expose by dragging it to the appropriate winlet. Thus, each winlet represents a particular property to expose. The views created by users can be named and are then available in the workspace designer for future (re)use. They can be copied as a starting point for a new view or used as a sub-view. The only semantics the designer understands are that property values can be either literals or resources – the base assumption in RDF. However, even this simple capability becomes powerful, when combined with the power to reuse existing views in creating new views. For literal properties, the user may specify whether or not the property is editable (i.e., an edit box is shown). Resource valued properties allow selection of an appropriate view to use to display it. If more than one instance of the property exists for the underlying entity, all the values are shown.

6 Discussion

Coupled with the relative simplicity and power of expressing information in arbitrary domains using RDF and the view abstraction in Haystack, the aforementioned tools situated within Haystack make significant gains toward our goal of empowering users to uniquely conceive an information-based task on the Semantic Web and build a corresponding workspace.

We have argued that our tools should allow building task specific workspaces for arbitrary tasks in any domain. A good test of their power then, would be to see how we could have used them to build workspaces for the various tasks involved in creating channels, views and workspaces. In fact, the tools we have discussed so far were indeed implemented as workspaces; they aggregated the necessary information resources for building channels, views and workspaces. (Look closely again at the channel manager and

note that it consists of winlets showing various collections of information that are relevant to the channel building task, e.g., query primitives, information types, etc. Fig. 5 shows the same channel manager's design view.)

The workspaces that users can design feature several desirable properties. For example, the navigate operation in Haystack can be curried with a destination workspace as a parameter, and added to a workspace to allow users to navigate between related tasks, much as in the Rooms project. More generally, the default Haystack single click behavior of navigating to the underlying entity when it is rendered using a particular type of view can be used to easily switch to a different workspace or information entity by simply adding a collection of such items to any workspace, and clicking on them as necessary. Also, the workspaces are not static, requiring developers to provide the functionality users need; users can change the workspace to meet changing requirements to include, remove or reuse information in multiple tasks immediately. Nevertheless, the various tools support significant extensibility by developers to either provide new query primitives, condition tests, layout engines, views or view designers. The initial setup time that the user invests is amortized over the numerous times he or she returns to the task, and all resources are readily available in order to immediately be productive. Thus, users can capture and transform what used to be a process of using multiple applications, into a single workspace application. Finally, since the workspaces themselves constitute information entities captured in RDF, they can easily be serialized and shared with others, thereby further amortizing the initial time investment.

Haystack also provides a number of benefits which we discuss below. The RDF data model immediately enables it to model information in multiple domains as well as interoperate with the Semantic Web by default. Additionally, since all system components are written in Adenine which itself can be compiled down to RDF, the system implementation itself is simply data that can be manipulated by the user. Thus, when users are changing the data in a workspace, they are reconfiguring Haystack's programming, in effect (re)programming it. By default, Haystack employs a single blackboard style store that allows content shared across multiple workspaces to remain consistent. Pervasive support for context menus and drag-and-drop facilitate a uniform interaction modality to all task workspaces.

We feel that the notion of channels as persistent queries whose result set is always current will be an important one as users are forced to deal with more information on the Semantic Web. The modularity of channels lets users define information properties in a virtual manner ahead of time, without knowing what they will apply to. This is possible, because the nature of the channel's content is known *a priori* based on the channel's query description. Thus, each individual item need not be annotated with certain properties; the fact that an item has certain properties that allows it to be a member of a channel, also allows it to (virtually) "inherit" the properties of the channel dynamically. Modularity also allows information channels to be reused in different contexts and redirected to different portions of the UI (within, or outside a task workspace) where the corresponding subset of information is useful. Or, the information corresponding to a channel may be redirected to a different device altogether, e.g. if an employee becomes sick or seeks to work from home, the channels appropriate to the work project can be subscribed to from the home computer. Channels as an abstraction are also useful in hiding the distributed and segmented nature of information by allowing aggregation of information from multiple stores. For example, the notion of viewing e-mails related to a particular topic regardless of which e-mail account it may have arrived in is a powerful one. As an indicator of the current user task focus, channels allow an information management platform a simple but useful technique to perform gate-keeping actions by minimizing users' interruption with events or information unrelated to the current task. For example, a user working on a task requiring information from a set of channels need not be interrupted by newly arrived (or created) information that does not fall into any of the channels.

Having implemented the simple view designer, we recognized that more powerful views can be designed by view designers that better understand the domain specific semantics of the underlying information that the views they generate will be manipulating. We advocate giving users the power to create their own views using appropriate view designers that interpret the underlying information using various semantics and can expose appropriate primitives for creating corresponding views. Semantics can be leveraged in various ways by view designers, while preserving valid data. For example, semantics used to interpret information can be used to allow users to choose between styles of views, e.g., a list of genetic bases (Adenine, Guanine, Thymine, Cytosine) can be interpreted appropriately by a view designer to allow the user to specify a preference of whether just the genetic sequence, or its complement is also to be rendered. Unlike the metadata lens view designer which uses a tiled layout and simply exposes property values for literal properties, such a view designer could allow users to select domain-specific preferences

such as laying out the genetic bases simply in a horizontal line as opposed to a helix, the colors to use for the genetic bases, etc.

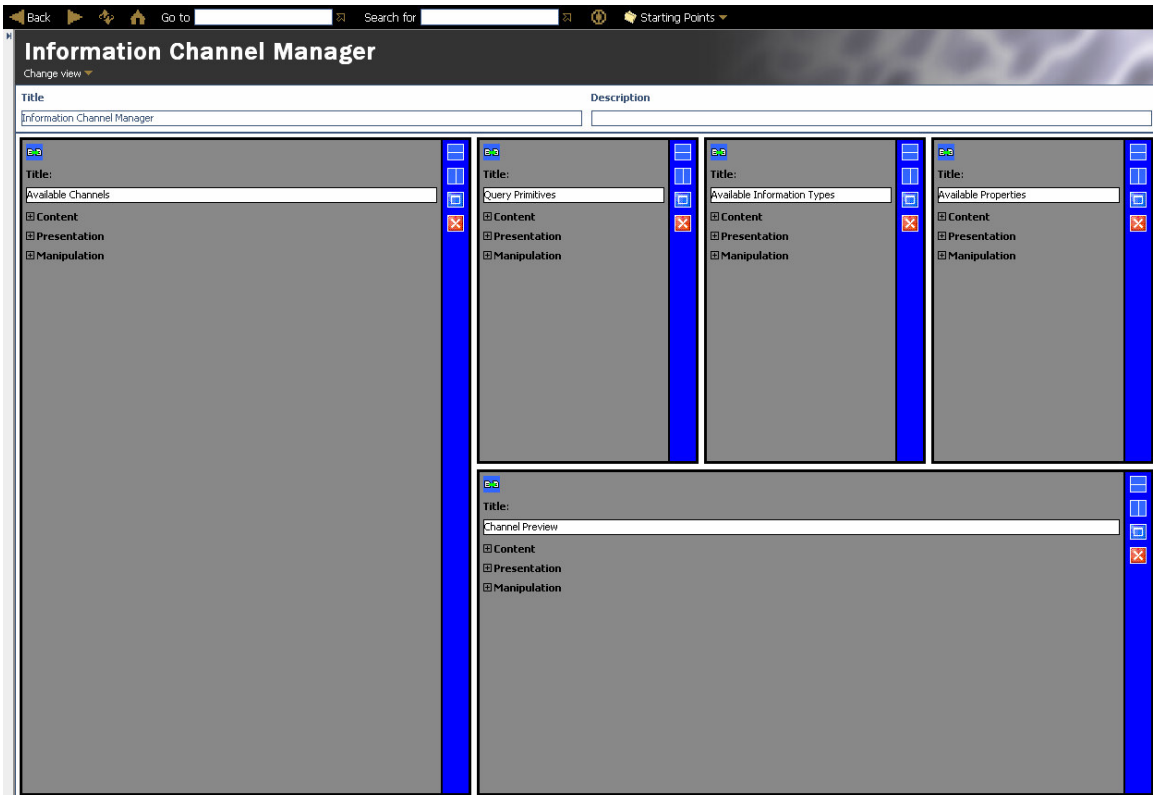


Fig. 5. Design View for the Information Channel Manager workspace

7 Conclusion & Future Work

In this paper, we have proposed an approach to information management that avoids the rigidity of domain specific applications and empowers the users to create flexible task workspaces that can take advantage of information on the Semantic Web immediately, without having to wait for a corresponding application to be developed. We have identified task workspaces as simultaneously an application of the Semantic Web, as well as a solution to information management problems it will complicate. In addition, we have attempted to identify three ingredients of workspaces that, given sufficient customization control over, users can employ effectively in building these workspaces to increase their productivity and minimize information overload. Furthermore, we recognize that these aspects of workspaces can be captured in reusable fragments: channels as units of content, views as units of presentation, and operations as units of manipulation. Whereas many of their constituent ideas and implementation techniques are not new, we believe that the tools discussed above are unique in that they embody all of them simultaneously. We believe that such tools that help people use the information available on the Semantic Web will be an enabling technology critical to its widespread adoption.

Our future work will concentrate on evaluating the utility of our tools via user studies, further investigating and refining the customizing capabilities that users need in building their workspaces, as well determining the best techniques to make such capabilities available (e.g., better UIs for querying RDF, other usability improvements, etc.). In addition, we hope to investigate other enabling infrastructure technologies that we anticipate will be required to make the Semantic Web a success and support task workspaces, including ontology translation servers that can translate between competing ontologies and prevent ontological “islands,” as well as view servers that can be used to share views that users have built for manipulating various information types.

8 Acknowledgements

This study was supported, in part, by the Biomedical Informatics Research Network (www.nbirn.net), Morphometry BIRN Testbed: 5U24RR021382.

References

1. Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web. *Scientific American*, May 2001.
2. http://xwinman.org/
3. Kandogan, E., Schneiderman, B. Elastic windows: Evaluation of multi-window operations. CHI 1997, pp. 250-257.
4. Hutchings, D. and Stasko, J. QuickSpace: New Operations for the Desktop Metaphor. Extended Abstracts of the Conference on Human Factors in Computing Systems 2002.
5. Card, S. and Henderson, D. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *ACM Transactions on Graphics* 5(3), July 1986, pp. 211-243.
6. Quan, D., Huynh, D., Karger, D. and Miller, R. User Interface Continuations. Proceedings of User Interface Software and Technology (UIST) 2003.
7. Myers, B. A. A Taxonomy of Window Manager User Interfaces. *IEEE Computer Graphics and Applications*, 8(5), September 1988, pp. 65-84.
8. Bellotti, V., Ducheneaut, N., Howard, M. and Smith, I. Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool. Proceedings of the Conference on Human Factors in Computing Systems 2003.
9. Kubi Software. www.kubisoftware.com
10. Anderson, C. and Horvitz, E. Web Montage: A Dynamic Personalized Start Page. Proceedings of the Eleventh International Conference on the World Wide Web, 2002.
11. Microsoft Office Online. http://office.microsoft.com.
12. *MyYahoo!* http://my.yahoo.com.
13. Schraefel, M., and Zhu, Y. Hunter Gatherer: A Collection Making Tool for the Web. Extended Abstracts of the Conference on Human Factors in Computing Systems 2002.
14. Cruz, I., and Lucas, W. A Visual Approach to Multimedia Querying and Presentation. Proceedings of the fifth ACM International Conference on Multimedia, 1997.
15. North, C. and Shneiderman, B. Snap-together visualization: A User Interface for Coordinating Visualizations via Relational Schemata. Proceedings of the Working Conference on Advanced Visual Interfaces, 2000, p.128-135.
16. Tan, D.S., Meyers, B. and Czerwinski, M. WinCuts: Manipulating Arbitrary Window Regions for More Effective Use of Screen Space. Extended Abstracts on Human Factors in Computing Systems, CHI 2004.
17. Hogue, A. and Karger, D. Wrapper Induction for End-User Semantic Content Development. Proceedings of First International Workshop on Interaction Design and the Semantic Web, ISWC 2004.
18. Rutledge, L., Houben, G. and Frasincar, F. Combining Generality and Specificity in Generating Hypermedia Interfaces for Semantically Annotated Repositories. Proceedings of First International Workshop on Interaction Design and the Semantic Web, ISWC 2004.
19. Missikoff, M., Navigli, R. and Velardi, P. The Usable Ontology: An Environment for Building and Assessing a Domain Ontology. Proceedings of ISWC 2002, p. 39.
20. Sure, Y., Erdmann, M., Angele, J., et al. OntoEdit: Collaborative Ontology Development for the Semantic Web. Proceedings of ISWC 2002, p. 221.
21. Dourish, P., Edwards, W. K., LaMarca, A., Salisbury, M. Presto: An Experimental Architecture for Fluid Interactive Document Spaces. *ACM Transactions on Computer-Human Interaction* (TOCHI), v.6 n.2, June 1999, pp.133-161.
22. Sicilia, M. and Garcia, E. Interaction Design of Ontology-Based Adaptive Resource Browsers Based on Selection. Proceedings of First International Workshop on Interaction Design and the Semantic Web, ISWC 2004.
23. Quan, D. and Karger, D. How to Make a Semantic Web Browser. Proceedings of WWW 2004.
24. Quan D., Karger, D. and Huynh, D. RDF Authoring Environments for End Users. Proceedings of the International Workshop on Semantic Web Foundation and Application Technologies, 2003.
25. Huynh, D., Karger, D. and Quan, D. Haystack: A Platform for Creating, Organizing, and Visualizing Information Using RDF. The Eighteenth National Conference on Artificial Intelligence, Workshop on Ontologies and the Semantic Web.
26. http://simile.mit.edu/welkin/
27. http://simile.mit.edu/longwell/
28. Quan, D., Huynh, D. and Karger, D. Haystack: A Platform for Authoring End User Semantic Web Applications. International Semantic Web Conference, 2003.
29. Quan, D. and Karger, D. Haystack: Metadata-Enabled Information Management. UIST, 2003.
30. Microsoft Visual Studio. http://msdn.microsoft.com/vstudio/
31. The Internet Brain Volume Database. http://www.cma.mgh.harvard.edu/ibvd/

Engineering a Semantic Desktop for Building Historians and Architects

René Witte[1], Petra Gerlach[2], Markus Joachim[3],
Thomas Kappler[1], Ralf Krestel[1], and Praharshana Perera[1]

[1] Institut für Programmstrukturen und Datenorganisation (IPD)
Universität Karlsruhe (TH), Germany
Email witte@ipd.uka.de
[2] Institut für Industrielle Bauproduktion (IFIB)
Universität Karlsruhe (TH), Germany
[3] Lehrstuhl für Denkmalpflege und Bauforschung
Universität Dortmund, Germany

Abstract. We analyse the requirements for an advanced semantic support of users—building historians and architects—of a multi-volume encyclopedia of architecture from the late 19^{th} century. Novel requirements include the integration of content retrieval, content development, and automated content analysis based on natural language processing.
We present a system architecture for the detected requirements and its current implementation. A complex scenario demonstrates how a desktop supporting semantic analysis can contribute to specific, relevant user tasks.

1 Introduction

Nowadays, information system users can access more content than ever before, faster than ever before. However, unlike the technology, the users themselves have not scaled up well. The challenge has shifted from finding information in the first place to actually locating useful knowledge within the retrieved content.

Consequently, research increasingly addresses questions of knowledge management and automated semantic analysis through a multitude of technologies [12], including ontologies and the semantic web, text mining and natural language analysis. Language technologies in particular promise to support users by automatically scanning, extracting, and transforming information from vast amounts of documents written in natural languages.

Even so, the question exactly how text mining tools can be incorporated into today's desktop environments, how the many individual analysis algorithms can contribute to a semantically richer understanding within a complex user scenario, has so far not been sufficiently addressed.

In this paper, we present a case study from a project delivering semantic analysis tools to end users, building historians and architects, for the analysis of a historic encyclopedia of architecture. A system architecture is developed upon a detailed analysis of the users' requirements. We discuss the current implementation and state first results from an ongoing evaluation.

2 Analyzing a Historical Encyclopedia of Architecture

Our ideas are perhaps best illustrated within the context of two related projects analysing a comprehensive multi-volume encyclopedia of architecture written in German in the late 19^{th} and early 20^{th} century.[4] In the following, we briefly outline the parties involved and motivate the requirements for an advanced semantic support of knowledge-intensive tasks, which are then presented in the next subsection.

The Encyclopedia. In the 19^{th} century the "Handbuch der Architektur" *(Handbook on Architecture)* was probably not the only but certainly the most comprehensive attempt to represent the entire, including present and past, building knowledge [6]. It is divided into four parts: Part I: Allgemeine Hochbaukunde *(general building knowledge)*, Part II: Baustile *(architectural styles)*, Part III: Hochbau-Konstruktionen *(building construction)*, and Part IV: Entwerfen, Anlage und Einrichtung der Gebäude *(design, conception, and interior of buildings)*.

Overall, it gives a detailed and comprehensive view within the fields of architectural history, architectural styles, construction, statics, building equipment, physics, design, building conception, and town planning.

But it is neither easy to get a general idea of the encyclopedia nor to find information on a certain topic. The encyclopedia has a complex and confusing structure: For each of its parts a different number of volumes—sometimes even split into several books—were published, all of them written by different authors. Some contain more than four hundred pages, others are much smaller, very few have an index. Furthermore, many volumes were reworked after a time and reprinted and an extensive supplement part was added. So referring to the complete work we are dealing with more than 140 individual publications and approximately at least 25 000 pages.

It is out of this complexity that the idea was born to support users—building historians and architects—in their work through state-of-the-art semantic analysis tools on top of classical database and information retrieval systems. However, in order to be able to offer the right tools we first needed to obtain an understanding on precisely what questions concern our users and how they carry out their related research.

User Groups: Building Historians and Architects. Two user groups are involved in the analysis within our projects: Building historians and architects. Those two parties have totally different perceptions of the "Handbuch der Architektur" and different expectations of its analysis. The handbook has got a kind of hybrid significance between its function as a research object and as a resource for practical use, between research and user knowledge.

An architect is planning, designing, and overseeing a building's construction. Although he is first of all associated with the construction of new buildings, more than 60% of building projects are related to the existing building stock, which

[4] Edited by Joseph Durm (⋆14.2.1837 Karlsruhe, Germany, †3.4.1919 ibidem) and three other architects since 1881.

means renovation, restoration, conversion, or extension of an existing building. For those projects he requires detailed knowledge about historic building construction and building materials or links to specialists skilled in this field. For him the gained information is not of scientific but of practical interest.

One of the specialists dealing with architecture from scientific motives is the building historian. All architecture is both the consequence of a cultural necessity and a document that keeps historical information over centuries. It is the task of architectural historians, building archaeologists, and art historians to decipher that information. Architectural history researches all historical aspects of design and construction regarding function, type, shape, material, design, and building processes. It is also considering the political, social, and economical aspects, the design process, the developments of different regions and times, the meaning of shape and its change throughout history. In order to "understand" an ancient building's construction and development, the building historian requires information about historical building techniques and materials. But he is also interested in the information sources themselves, in their history of origin, their development, and their time of writing. Literature research is one of his classical tools.

2.1 Requirements Analysis

We now examine the requirements for a semantic desktop support; first, from a user's perspective, and second, their technical consequents.

User Requirements. For the building historian the handbook itself is object and basis of his research. He puts a high value on a comprehensible documentation of information development, since the analysis and interpretation of the documentation process itself is also an important part of his scientific work. The original text, the original object is the most significant source of cognition for him. All amendments and notes added by different users have to be managed on separate annotation or discussion levels—this would be the forum for scientific controversy, which may result in new interpretations and cognition.

For the architect the computer-aided analysis and accessibility of the encyclopedia is a means to an end. It becomes a guideline offering basic knowledge of former building techniques and construction. The architect is interested in technical information, not in the process of cognition. He requires a clearly structured presentation of all available information on one concept. Besides refined queries ("semantic queries") he requires further linked information, for example web sites, thesauruses, DIN and EU standards, or planning tools.

Both user groups are primarily interested in the content of the encyclopedia, but also in the possibility of finding "unexpected information,"[5] as this would afford a new quality of reception. So far it is not possible to conceive this complex and multi-volume opus with thousands of pages at large: The partition of the handbook in topics, volumes, and books is making the retrieval of a particular

[5] Information delivered through a user's desktop is termed *unexpected* when it is relevant to the task at hand yet not explicitly requested.

concept quite complicated. Only the table of contents is available to give a rough orientation. But it's impossible to get any information about single concepts or terms. You can neither find an overall index nor—apart from a few exceptions—an index of single volumes. Because each of them comprises a huge amount of text, charts, and illustrations, it is unlikely to find the sought-for term coincidentally by running over the pages. Thus, this project's aim is to enable new possibilities of access by the integration of "semantic search engines" and automated analyses. An automated index generation alone would mean a substantial progress for further research work.

System Requirements. In [10] we previously examined the requirements for a system architecture supporting knowledge-intensive tasks, like the ones stated in our case study. Its most important conclusion is that such a system needs to integrate the classically separated areas of information retrieval, content development, and semantic analysis.

Information Retrieval. The typical workflow of a knowledge worker starts by retrieving relevant information. IR systems support the retrieval of documents from a collection based on a number of keywords through various search and ranking algorithms [1,5]. However, with a large number of relevant documents (or search terms that are too broad) this so-called "bag of words approach" easily results in too many potentially relevant documents, leading to a feeling of "information overload" by the user. Furthermore, the retrieval of documents is no end in itself: Users are concerned with the development of new content (like reports or research papers) and only perform a manual search because current systems are not intelligent enough to sense a user's need for information and proactively deliver relevant information based on his current context.

Thus, while also offering our users the classical full-text search and document retrieval functions, we must additionally examine a tighter integration with content development and analysis tools.

Content Development. New content is developed by our users through a number of tasks as outlined above: from questions and notes arising from the examination of a specific building through interdisciplinary discussions to formal research papers. At the same time, access to existing information, like the handbook, and previous results is needed, preferably within a unified interface.

As a model for this mostly interactive and iterative process we propose to employ a *Wiki* system [7], as they have proven to work surprisingly well for cooperative, decentralized content creation and editing. Traditionally, Wikis have been used to develop new material, but our approach here is to combine both existing and new content within the same architecture by integrating (and enhancing) one of the freely available Wiki engines.

Wiki systems allow us to satisfy another requirement, namely the users' need to be able to add their own information to a knowledge source; for example, a building historian might want to add a detailed analysis to a chapter of the encyclopedia, while an architect might want to annotate a section with experiences

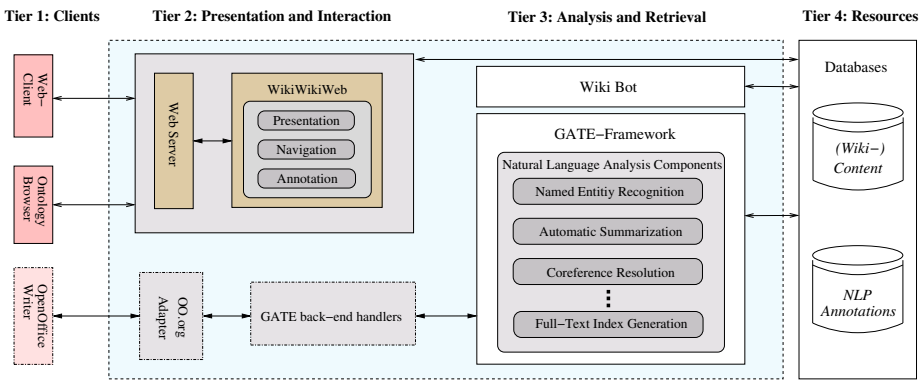


Fig. 1. Architecture integrating content development, retrieval, and analysis

gathered from the restoration of a specific building. Wiki systems typically offer built-in discussion and versioning facilities matching these requirements.

Semantic Analysis. Automated semantic analysis will be provided through tools from the area of natural language processing (NLP), like text mining and information extraction. Typical NLP tasks, which we will discuss in more detail below, are document classification and clustering, automatic summarization, named entity recognition and tracking, and co-reference resolution.

The aforementioned integration of information retrieval, content development, and analysis allows for new synergies between these technologies: content in the Wiki can be continually scanned by NLP pipelines, which add their findings as annotations to the documents for user inspection and internal databases for later cross-reference. When a user now starts to work on a new topic, e.g., by means of creating a new Wiki page, the system can analyse the topic and pro-actively search and propose relevant entities from the database to the user.

3 System Architecture

We now present the architecture we developed to support the detected requirements, as it is currently being implemented. It is based on the standard multi-tier information system design (Fig. 1). Its primary goal is to integrate document retrieval, automated semantic analysis, and content annotation as outlined above. We now discuss each of the four tiers in detail.

Tier 1: Clients. The first tier provides access to the system, typically for humans, but potentially also for other automated clients. A web browser is the standard tool for accessing the Wiki system. Additional "fat" clients, like an ontology browser, are also supported. The integration of the OpenOffice.org word processor is planned for a future version.

Tier 2: Presentation and Interaction. Tier 2 is responsible for information presentation and user interaction. In our architecture it has to deal with both content

development and visualization. In the implementation, most of the functionality here is provided through standard open source components, like the *Apache* web server and the *MediaWiki*[6] content management system.

Tier 3: Retrieval and Analysis. Tier 3 provides all the document analysis and retrieval functionalities outlined above. In addition to the search facilities offered by the Wiki system, a database of NLP annotations (e.g, named entities) can be searched through the *Lucene*[7] search engine.

Semantic analysis of texts through natural language processing (NLP) is based on the GATE framework, which we will discuss in Section 4.3.

The results of the automatic analyses are made visible in an asynchronous fashion through the Wiki system, either as individual pages, or as annotations to existing pages. Thus, automatically created analysis results become first-class citizens: Original content, human, and machine annotations constitute a combined view of the available knowledge, which forms the basis for the cyclic, iterative create-retrieve-analyse process outlined above.

Tier 4: Resources. Resources (documents) either come directly from the Web (or some other networked source, like emails), or a full-text database holding the Wiki content. The GATE framework provides the necessary resource handlers for accessing texts transparently across different (network) protocols.

4 Implementation

In this section we highlight some of the challenges we encountered when implementing the architecture discussed above, as well as their solutions.

4.1 Digitizing History

For our project most of the source material, especially the historical encyclopedia, arrived in non-digital form. As a first step, the documents had to be digitized using specialized book scanners, which were available through Universität Karlsruhe's main library. For automatic document processing, however, scanned page images are unusable. Unfortunately, due to the complexity of the encyclopedia's layout (including diagrams, formulas, tables, sketches, photos, and other formats) and the inconsistent quality of the 100-year old source material, automatic conversion via OCR tools proved to be too unreliable. As we did not want to engage in OCR research, a manual conversion of the scanned material into an electronic document was the fastest and most reliable option that preserved the original layout information, such as footnotes, chapter titles, figure captions, and margin notes. This task was outsourced to a Chinese company for cost reasons.

4.2 Information Storage and Retrieval Subsystem

The encyclopedia is made accessible via *MediaWiki* [9], which is a popular open source Wiki system best known for its use within the *Wikipedia*[8] projects. Media-

[6] http://www.mediawiki.org
[7] http://lucene.apache.org
[8] http://www.wikipedia.org

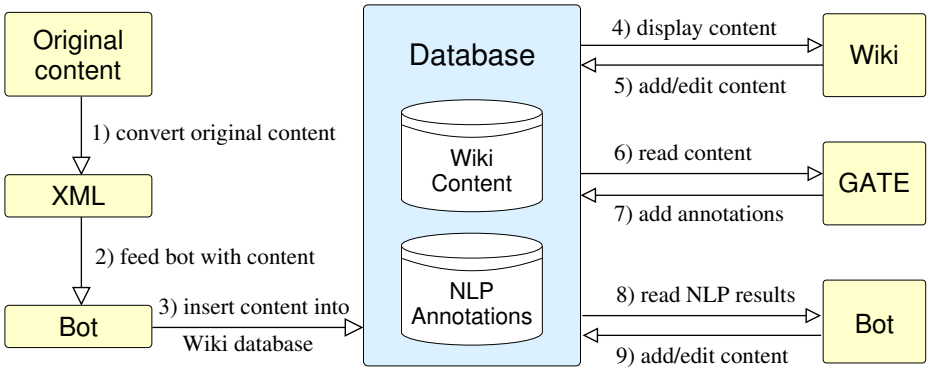


Fig. 2. Workflow between document storage, retrieval, and NLP analysis

Wiki stores the textual content in a MySQL database, the image files are stored as plain files on the server. It provides a PHP-based dynamic web interface for browsing, searching, and manual editing of the content.

The workflow between the Wiki and the NLP subsystems is shown in Fig. 2. The individual sub-components are loosely coupled through XML-based data exchange. Basically, three steps are necessary to populate the Wiki with both the encyclopedia text and the additional data generated by the NLP subsystem. These steps are performed by a custom software system written in Python.

Firstly (step (1) in Fig. 2), the original *Tustep*[9] markup of the digitized version of the encyclopedia is converted to XML. The resulting XML intends to be as semantically close to the original markup as possible; as such, it contains mostly layout information. It is then possible to use XSLT transformations to create XML that is suitable for being processed in the natural language processing (NLP) subsystem described below.

Secondly (2), the XML data is converted to the text markup used by MediaWiki. The data is parsed using the Python `xml.dom` library, creating a document tree according to the W3C DOM specification.[10] This allows for easy and flexible data transformation, e.g., changing an element node of the document tree such as `<page no="12">` to a text node containing the appropriate Wiki markup.

And thirdly (3), the created Wiki markup is added to the MediaWiki system using parts of the Python Wikipedia Robot Framework,[11] a library offering routines for tasks such as adding, deleting, and modifying pages of a Wiki or changing the time stamps of pages. Fig. 3 shows an example of the converted end result, as it can be accessed by a user.

While users can (4) view, (5) add, or modify content directly through the Wiki system, an interesting question was how to integrate the NLP subsystem, so that it can read information (like the encyclopedia, user notes, or other pages) from the Wiki as well and deliver newly discovered information back to the users.

[9] http://www.zdv.uni-tuebingen.de/tustep/tustep_eng.html
[10] http://www.w3.org/DOM/
[11] http://pywikipediabot.sf.net

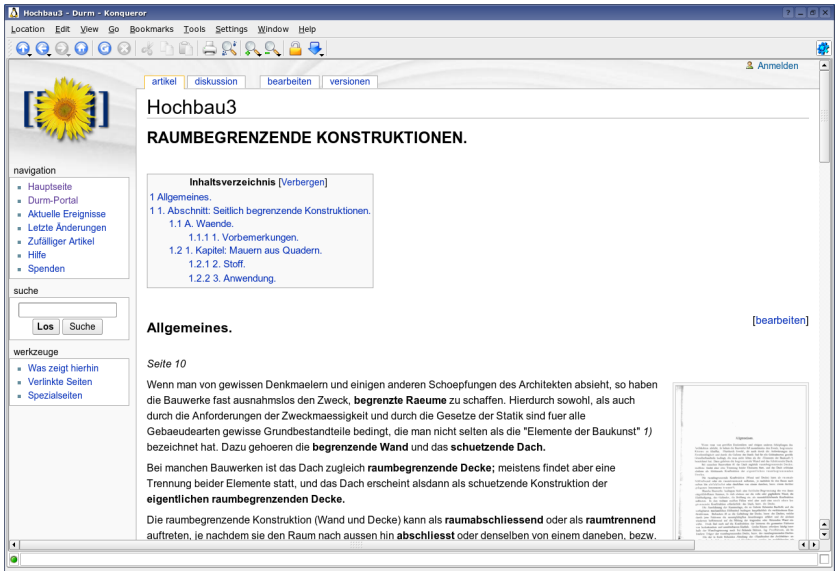


Fig. 3. Content from the encyclopedia accessible through *MediaWiki*

Our solution for this is twofold: for the automated analysis, we asynchronously run all NLP pipelines (described in Section 4.3) on new or changed content (6). The results are then (7) stored as annotations in a database.

The Wiki bot described above is also responsible for adding results from the natural language analysis to the Wiki. It asynchronously (8) reads new NLP annotations and (9) adds or edits content in the Wiki database, based on templates and namespaces. NLP results can appear in the Wiki in two forms: as new individual pages, or within the "discussion section" connected to each page through a special namespace convention within the MediaWiki system. Discussion pages were originally introduced to hold meta-information, like comments, additions, or questions, but we also use them for certain kinds of NLP results, like storing automatically created summaries for the corresponding main page. Other information generated by the NLP subsystem, such as the automatic index generation detailed in Section 4.3, are added to the Wiki as separate pages.

4.3 NLP Subsystem

The natural language analysis part is based on the GATE *(General Architecture for Text Engineering)* framework [4], one of the most widely used NLP tools. Since it has been designed as a component-based architecture, individual analysis components can be easily added, modified, or removed from the system.

A document is processed by a sequential *pipeline* of processing components. These pipelines typically start with basic preprocessing components, like tokenization, and build up to more complex analysis tasks. Each component can add (and read previous) results to the text in form of *annotations*, which form a graph over the document, comparable to the TIPSTER annotation model.

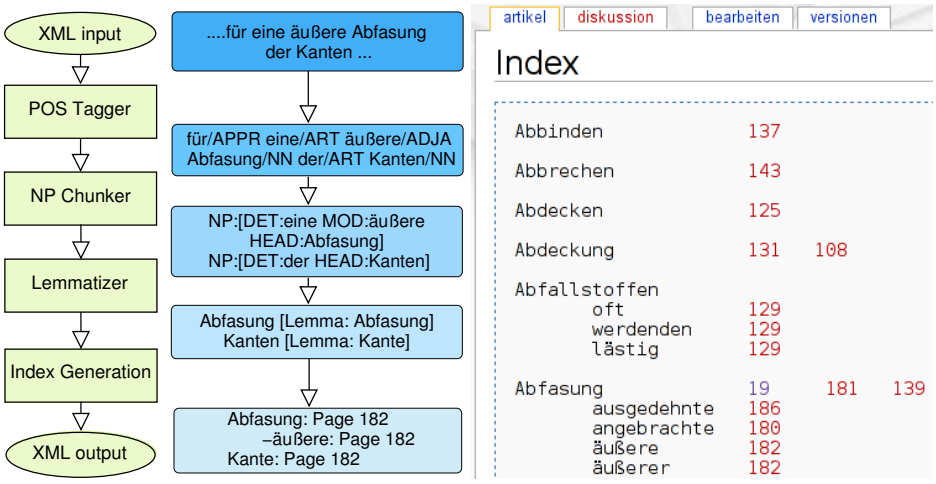


Fig. 4. NLP pipeline for the generation of a full-text index (left side) and its integration into the Wiki system (right side)

We now discuss some the NLP pipelines currently in use; however, it is important to note that new applications can easily be assembled from components and deployed within our architecture.

Automatic Index Generation. Many documents do not come with a classical full-text index, which significantly hinders access to the contained information. Examples include collections of scientific papers, emails, and within our project especially the historical encyclopedia.

In order to allow easier access to the contained information, we use our language tools to automatically create a full-text index from the source documents. This kind of index is targeted at human users and differs from classical indexing for information retrieval in that it is more linguistically motivated: only so-called *noun phrases* (NPs) are permitted within the index, as they form the grammatical base for named entities (NEs) identifying important concepts.

Index generation is implemented as a processing component in the NLP pipeline, which builds upon the information generated by other language components, particularly a part-of-speech (POS) tagger, an NP chunker, and a context-aware lemmatizer (see [8] for details on these steps).

For each noun phrase, we track its lemma (uninflected form), modifiers, and page number. Nouns that have the same lemma are merged together with all their information. Then, we create an inverted index with the lemma as the main column and their modifiers as sub-indexes (Fig. 4, left side).

The result of the index generation component is another XML file that can be inserted into the Wiki system through the Framework described above. Fig. 4 (right side) shows an excerpt of the generated index page for the encyclopedia.

Automatic Context-Based Summarization. Automatically generated summaries are condensed derivatives of a single or a collection of source text(s),

reducing content by selection and/or generalisation on what is important. Summaries serve an indicative purpose: they aim to help a time-constrained human reader with the decision whether he wants to read a certain document or not.

The state of the art in automatic summarization is exemplified by the yearly system competition organized by NIST within the Document Understanding Conference (DUC) [3]. Our summarization pipeline is based on the ERSS system that participated in the DUC competitions from 2003–2005, with some modifications for the German language. One of its main features is the use of fuzzy set theory to build coreference chains and create summaries [11], which enables the user to set thresholds that directly influence the granularity of the results. For more details on the system and its evaluation, we refer the reader to [2]. Summaries can take various forms:

Single-Document Summaries. A single-document summary can range from a short, headline-like 10-word keyword list to multiple sentences or paragraphs. We create these summaries for individual Wiki pages (e.g., holding a chapter of the handbook) and attach the result to its corresponding discussion page.

Multi-Document Summaries. For longer documents, made up of various sections or chapters, or whole document sets, we perform multi-document summarization. The results are stored as new Wiki pages and are typically used for content-based navigation through a document collection.

Focused and Context-Based Summaries. This most advanced form of multi-document summarization does not create summaries in a generic way but rather based on an explicit question or user context. This allows for the pro-active content generation outlined above: a user working on a set of questions, stated in a Wiki page (or, in future versions, simple by typing them into a word processor), implicitly creates a context that can be detected by the NLP subsystem and fed into the context-based summarization pipeline, delivering content from the database to the user that contains potentially relevant information. We show an example in Section 5.2.

Ontology-based Navigation and Named Entity Detection. Ontologies are a more recent addition to our system. We aim to evaluate their impact on the performance of the named entity (NE) recognition, as well as semantic navigation through a browser.

Named entities are instances of concepts. They are particular to an application domain, like *person* and *location* in the newspaper domain, *protein* and *organism* in the biology domain, or *building material* and *wall type* in the architecture domain.

The detection of named entities is important both for users searching for particular occurrences of a concept and higher-level NLP processing tasks. One way of detecting these NEs, supported by the GATE framework, is a markup of specific words, defined in *Gazetteer* lists, which can then be used together with other grammatical analysis results in so-called finite-state transducers defined through regular-expression-based grammars in the JAPE language.[12]

[12] For more details, please refer to the GATE user's guide: `http://gate.ac.uk/`.

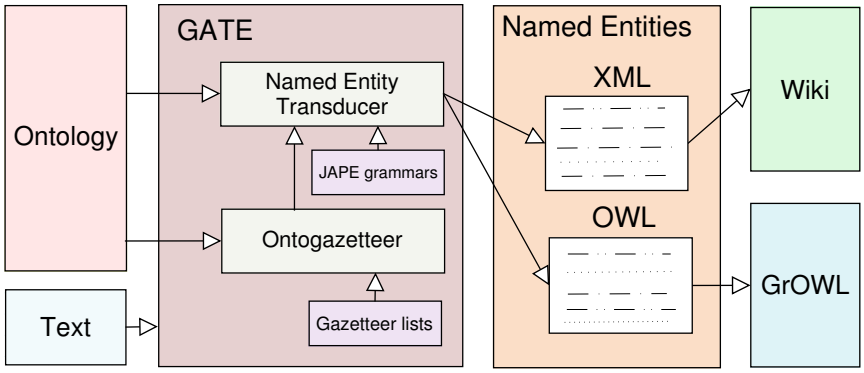


Fig. 5. Ontology-aware named entity detection through gazetteers and finite-state transducers delivering results in various output formats

The addition of ontologies (in DAML format) allows to locate entities within an ontology (currently, GATE only supports taxonomic relationships) through ontology extensions of the Gazetteer and JAPE components. The detected entities are then exported in an XML format for insertion into the Wiki and as an OWL RDF file (Fig. 5).

NE results are integrated into the Wiki similarly to the index system described above, linking entities to content pages. The additional OWL export allows for a graphical navigation of the content through an ontology browser like GrOWL.[13] The ontologies exported by the NLP subsystem contain sentences as another top-level concept, which allows to navigate from domain-specific terms directly to positions in the document mentioning a concept, as shown in Fig. 6.

5 Evaluation

We illustrate a complex example scenario where a building historian or architect would ask for support from the integrated system.

5.1 Scenario

The iterative analysis process oriented on the different requirements of the two user groups is currently being tested on the volume "Wände und Wandöffnungen"[14] *(walls and wall openings)*. It describes the construction of walls, windows, and doors according to the type of building material. The volume has 506 pages with 956 figures; it contains a total of 341 021 tokens including 81 741 noun phrases.

Both user groups are involved in a common scenario: The building historian is analysing a 19^{th} century building with regard to its worth of preservation in order to be able to identify and classify its historical, cultural, and technical

[13] http://seek.ecoinformatics.org/Wiki.jsp?page=Growl

[14] E. Marx: *Wände und Wandöffnungen*. Aus der Reihe: Handbuch der Architektur. Dritter Teil, 2. Band, Heft I, 2. Auflage, Stuttgart 1900.

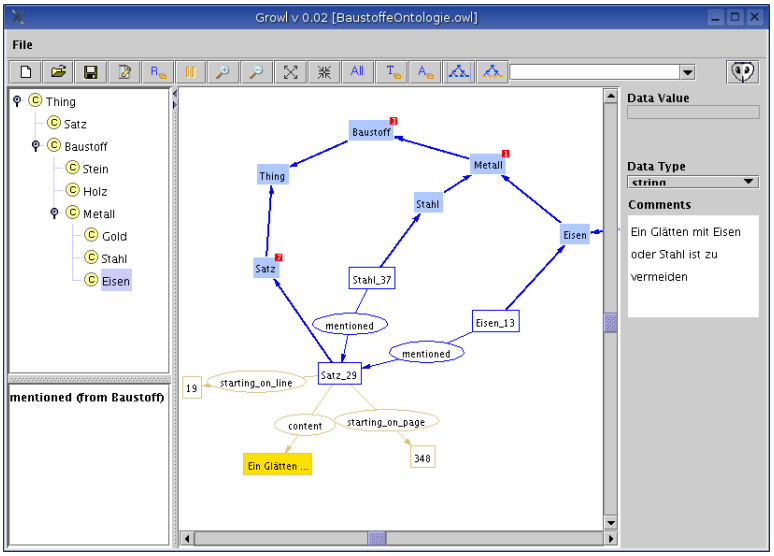


Fig. 6. Ontology excerpt on *building materials* visualized using GrOWL showing NLP-detected instances linked to concepts, anchored within the text

value. The quoins, the window lintels, jambs, and sills as well as door lintels and reveals are made of fine wrought parallelepipedal cut sandstones. The walls are laid of inferior and partly defective brickwork. Vestiges of clay can be found on the joint and corner zones of the brickwork. Therefore, a building historian could make the educated guess that the bricks had been rendered with at least one layer of external plaster. Following an inspection of the building together with a restorer, the historian is searching in building documents and other historical sources for references to the different construction phases. In order to analyse the findings it is necessary to become acquainted with plaster techniques and building materials. Appropriate definitions and linked information can be found in the encyclopedia and other sources. For example, he would like to determine the date of origin of each constructional element and whether it is original or has been replaced by other components. Was it built according to the state-of-the-art, does it feature particular details?

In addition, he would like to learn about the different techniques of plastering and the resulting surfaces as well as the necessary tools. To discuss his findings and exchange experiences he may need to communicate with other colleagues.

Even though he is dealing with the same building, the architect's aim is another. His job is to restore the building as carefully as possible. Consequently, he needs to become acquainted with suitable building techniques and materials, for example, information about the restoration of the brick bond. A comprehensive literature search may offer some valuable references to complement the conclusion resulting from the first building inspection and the documentation of the construction phases.

“Welche Art von Putz bietet Schutz vor Witterung?”
Ist das Dichten der Fugen für die Erhaltung der Mauerwerke, namentlich an den der Witterung ausgesetzten Stellen, von Wichtigkeit, so ist es nicht minder die Beschaffenheit der Steine selbst. Bei der früher allgemein üblichen Art der gleichzeitigen Ausführung von Verblendung und Hintermauerung war allerdings mannigfach Gelegenheit zur Beschmutzung und Beschädigung der Verblendsteine geboten. . . .

Fig. 7. Excerpt from a focused summary generated based on a question (shown on top), generated by the NLP subsystem through automatic summarization

5.2 Desktop Support

So far, we have been testing the desktop with the Wiki system and three integrated NLP tools within the project. We illustrate how our users ask for semantic support from the system within the stated scenario.

NLP Index. As the tested volume offers just a table of contents but no index itself, an automatically generated index is a very helpful and timesaving tool for further research: Now it is possible to get a detailed record on which pages contain relevant information about a certain term. And because the adjectives of the terms are indicated as well, information can be found and retrieved very quickly, e.g., the architect analysing the plain brickwork will search for all pages referring to the term “Wand” *(wall)* and in particular to “unverputzte Wand” *(unplastered wall).*

Summaries. Interesting information about a certain topic is often distributed across the different chapters of a volume. In this case the possibility to generate an automatic summary based on a context is another timesaving advantage. The summary provides a series of relevant sentences, e.g., to the question (Fig. 7): “Welche Art von Putz bietet Schutz vor Witterung?” *(Which kind of plaster would be suitable to protect brickwork against weather influences?).* An interesting properties of these context-based summaries is that they often provide “unexpected information,” relevant content that a user most likely would not have found directly.

The first sentence of the automatic summarization means: *The joint filling is important for the resistance of the brickwork, especially for those parts exposed to the weather, as well as the quality of the bricks.* This is interesting for our example because the architect can find in the handbook—following the link—some information about the quality of bricks. Now he may be able to realize that those bricks used for the walls of our 19^{th} century building are not intended for fare-faced masonry. After that he can examine the brickwork and will find the mentioned vestiges of clay.

The architect can now communicate his findings via the Wiki discussion page. After studying the same text passage the building historian identifies the kind of brickwork, possibly finding a parallel to another building in the neighborhood, researched one year ago. So far, he was not able to date the former building precisely because all building records have been lost during the war. But our

example building has a building date above the entrance door and therefore he is now able to date both of them.

Named Entity Recognition and Ontology-based Navigation. Browsing the content, either graphically or textually, through ontological concepts is another helpful tool for the users, especially if they are not familiar in detail with the subject field of the search, as it now becomes possible to approach it by switching to super- or subordinate concepts or instances in order to get an overview. For example, restoration of the windows requires information of their iron construction. Thus, a user can start his search with the concept "Eisen" *(iron)* in the ontology (see Fig. 6). He can now navigate to instances in the handbook that have been linked to "iron" through the NLP subsystem, finding content that mentions window and wall constructions using iron. Then he can switch directly to the indicated parts of the original text, or start a more precise query with the gained information.

5.3 Summary

The offered semantic desktop tools, tested so far on a single complete volume of the encyclopedia, turned out to be a real support for both our building historians and architects: Automatic indices, summaries, and ontology-based navigation can help them to find relevant, precisely structured and cross-linked information to certain, even complex topics in a quick and convenient fashion. The system's ability to cross-link, network, and combine content across the whole collection have the potential to guide the user to unexpected information, which he might not have realized even when completely reading the sources themselves.

In doing so the tools' time saving effects seems to be the biggest advantage: Both user groups can now concentrate on their research or building tasks—they do not need to deal with the time-consuming and difficult process of finding interesting and relevant information.

6 Conclusions and Future Work

In this paper, we showed how a user's desktop can integrate content retrieval, development, and NLP-based semantic analysis. The architecture is based on actual users' requirements and preliminary evaluations show the feasibility and usefulness of our approach. We believe our system also applies to other domains.

From a technical perspective, the biggest challenge is the lack of hooks in standard desktop components designed for use by humans enabling read-, write-, and navigate operations from automated components, requiring expensive workarounds. In our system, automated access to the Wiki system by the NLP subsystem requires the use of a bot, which was not originally designed for that purpose. We currently face similar problems integrating the OpenOffice.org word processor into the system. There is currently no way several desktop components can share a common semantic resource, like an ontology, or even delegate analysis tasks on behalf of a user. On a smaller scale, we are currently working on integrating a description logic (DL) reasoning system to allow semantic queries based on the automatically extracted entities.

However, one of the most interesting questions, from an information system engineer's standpoint, is a concern raised by our building historians: the apparent loss of knowledge throughout the years, which occurs when users of automated systems narrowly apply retrieved information without regard for its background, connections, or implications; or when they simply do not even find all available information because concepts and techniques have been lost over the years: As a result, a user might no longer be aware of existing knowledge because he lacks the proper terminology to actually retrieve it. While an analysis of this effect is still an ongoing consideration, we hope that the multitude of access paths offered by our integrated approach at least alleviates this problem.

Acknowledgments. The work presented here is funded by the German research foundation (DFG) through two related projects: "Josef Durm" (HA 3239/4-1, building history, Uta Hassler and KO 1488/7-1, architecture, Niklaus Kohler) and "Entstehungswissen" (LO296/18-1, informatics, Peter C. Lockemann).

References

1. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
2. Sabine Bergler, René Witte, Zhuoyan Li, Michelle Khalifé, Yunyu Chen, Monia Doandes, and Alina Andreevskaia. Multi-ERSS and ERSS 2004. In *Workshop on Text Summarization*, Document Understanding Conference (DUC), Boston Park Plaza Hotel and Towers, Boston, USA, May 6–7 2004. NIST.
3. Document Understanding Conference. http://duc.nist.gov/.
4. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the ACL*, 2002. http://gate.ac.uk.
5. Reginald Ferber. *Information Retrieval*. dpunkt.verlag, 2003.
6. Ulrike Grammbitter. *Josef Durm (1837–1919). Eine Einführung in das architektonische Werk*, volume 9 of *tuduv-Studien: Reihe Kunstgeschichte*. tuduv-Verlagsgesellschaft, München, 1984. ISBN 3-88073-148-9.
7. Bo Leuf and Ward Cunningham. *The Wiki Way, Quick Collaboration on the Web*. Addison-Wesley, 2001.
8. Praharshana Perera and René Witte. A Self-Learning Context-Aware Lemmatizer for German. In *Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, B.C., Canada, October 6–8 2005.
9. Wikipedia, the free encyclopedia. Mediawiki. http://en.wikipedia.org/wiki/MediaWiki; accessed July 26, 2005.
10. René Witte. An Integration Architecture for User-Centric Document Creation, Retrieval, and Analysis. In *Proceedings of the VLDB Workshop on Information Integration on the Web (IIWeb)*, pages 141–144, Toronto, Canada, August 30 2004.
11. René Witte and Sabine Bergler. Fuzzy Coreference Resolution for Summarization. In *Proc. of 2003 Int. Symposium on Reference Resolution and Its Applications to Question Answering and Summarization (ARQAS)*, pages 43–50, Venice, Italy, June 23–24 2003. Università Ca' Foscari. http://rene-witte.net.
12. Ning Zhong, Jiming Liu, and Yiyu Yao, editors. *Web Intelligence*. Springer, 2003.

Lessons for the future of Semantic Desktops learnt from 10 years of experience with the IDELIANCE Semantic Networks Manager

Jean Rohmer
CENTAI (Centre of Excellence For Information Analysis)
Thales Communications
146 Avenue de Valmy BP 82 92704 Colombes Cedex France
jean.rohmer@fr.thalesgroup.com

ABSTRACT

In 1993, we started to develop a tool to help people manage structured personal and business information. Initially named ORCCA (Online Resources for Corporate Citizens Action), it was fully implemented as a Semantic Networks Editor and Manager. Initially designed as a personal desktop tool, it evolved towards a collaborative server allowing small groups to share Semantic Networks. It has been marketed under the name of IDELIANCE, and has been used in significant business applications by a variety of users. During all these pioneer phases of design, implementation and usage of Ideliance, we learnt a lot of lessons which could be of interest for the success of the emerging Semantic Desktop domain. Not only did we implement practical tools for semantic networks management, but we also closely observed hundreds of people using Semantic Networks as a new way of writing and reading information. In this position paper, we propose a survey of the key points we encountered. For each of these points, we describe our experience, and we propose some guidelines for future Semantic Desktops design and usage.

Note : this paper is not a technical paper about Ideliance, or about its evolution towards the Semantic Web. It is a position paper which aims at expressing the general lessons we learned for the benefit of future designers and users.

HISTORY

History is important to position our work in its context so as to better reuse it in the current Semantic Web landscape.

In 1993, the so-called AI (Artificial Intellingence) *winter* was starting. In the 80's we had been deeply involved in the AI experience, as manager of a large products and services team (Groupe Bull Cediag), which achieved significant expert systems (like the SACHEM blast furnaces monitoring for ARCELOR) and innovative tools, like the first commercial implementation of constraint logical programming (CLP), CHARME. But a serious economical crisis made clear that these times were gone, and teams and projects disbanded. In the meantime, a new concept was emerging : « Knowledge Management » (KM), mainly boosted by new collaborative tools like Lotus Notes. Remember that, at this time the WEB simply did not exist in the Business and Industry world, even if the essential work had already been done by Tim Berners-Lee, and if Gopher and Mosaïc were close to a reality.
Thus we decided to design a KM tool by recycling the simplest and most universal part of AI –its *representation* features- as opposed to its *automatic reasoning* features. And, for the sake of generality and simplicity, the Semantic Networks paradigm was chosen. This led us to develop a first version of a *personal semantic network editor* during the extended week-end of november 11, 1993. We then created a small company to develop this idea. We were involved in two important projects which influenced strongly the evolution of IDELIANCE : (1) The Mnemos project – an european EUREKA project with, among others, Aerospatiale (now EADS), Matra, French Atomic Energy Commission and NFT-Raufoss from Norway. In this « Corporate Memory » project we were in charge of developping a « Personal Memory Assistant », (2) the OCTO project , for the French Ministry of Defence, whose acronym meant « Conceptual Objects of the Battlefield ». Its aim was to develop a tool to help headquarters officers represent complex situations and take decisions.
By 1998, we started to market a personal version of Ideliance to large companies, like L'Oreal, Air Liquide, France Telecom, and to the French Army. Following pressure from customers, we developped a client-server version of Ideliance, allowing small communities of persons to share semantic networks.
In 2002, lacking financial resources for its development, the Ideliance company had to stop, and the product was underpinned by Thales, a very large Defence and Electronics company, where it is mainly used for Security and Military Intelligence applications.
Main contributors of Ideliance were Sylvie Le Bars (www.arkandis.com) and myself for the design, and Stéphane Jean and Denis Poisson for the implementation.

There are few available academic references about Ideliance, because it was first developped in a SME, then its main applications were in the Defence domain. Some references are given at the end of this paper.

In some sense, Ideliance was a fruit of the AI winter, and made a bridge between the AI times and the current Semantic Web era. A deep knowledge in AI, Logic Programming, Knowledge Engineering is useful for the design of future desktop systems.

MAIN DESIGN PRINCIPLES

Principle 1) : Simplicity. The main idea is to let users write semantic networks for themselves, to express any information of interest in their activity. The tool must be simple enough to allow any person capable of structuring its thoughts to translate it into a semantic network. We call « collection » a given semantic network, following the rule that concepts and vocabulary to describe and use Ideliance should *absolutely not* refer to technical or theoretical terms of computer science. With the Ideliance editor, users can create *Subjects*, assign them to *Categories*, and link them through *Relations* to constitute *Statements* like *« Jim / works for / W3C »*. All relations have a reverse relation, so that the reverse Statement *« W3C / employs / Jim »* is built automatically. Subjects in statements can also be dates, files, URLs or mail addresses. Text notes can be attached to a subject. There are no type or cardinality constraints.

N.B. An intermediate name for Ideliance was *Idécriture.*

Principle 2) : Emergence. There is no a priori declaration of structures, models, templates, classes … All these notions emerge automatically from the reality of the semantic network : if you want to describe a new Person, Ideliance makes you suggestions computed from the current set of information : relations most frequently used to describe a person, and most frequently used subjects used to complement them are proposed as a first choice to the user. This principle of emergence, learning, discovery is used thoroughly in all aspects of the user interface.

Principle 3) : Powerful Query and Reporting tools : There is in Ideliance a plethora –although of various levels of interest and usage- of tools to ask queries, draw relations graphs, build dynamic OLAP reporting arrays, from the semantic network, and to publish it in various document formats.

Principle 4) : Gateways with usual data formats : Structured information from tables of spreadsheets and relational databases is easily translated into the semantic network format, and the other way round.

Principle of (Simplicity + Emergence) is a nice compromise. It lets novice user start with simple things, without the burden of a-priori modelling. It allows also to start with simple Semantic Desktop engines, which can be later improved by adding more sophisticated learning and discovery algorithms, in a smooth transition mode for the user.
Gateways with other formats is key to have users accept putting critical information in the Semantic Desktop. And Query and Reporting tools should compare with what they can get from tools like Access, Excel, Crystal Report. We experienced repeatedly that the main competitor for a Semantic Desktops is … MS Excel ! We will address the role of a Semantic Desktop vis à vis the Semantic Web in another point.

A key –yet difficult- point for the success of Semantic Desktops is to persuade users that it isn't more demanding in terms of effort and intelligence than using Excel, and that the benefits will increase tenfold.

APPLICATIONS / MARKET SEGMENTS

We found the following types of applications, which can be considered as market segments for Semantic Desktops in general :

--general purpose permanent personal information mamagement
Individuals decide by themselves to store and retrieve information strictly under the form of semantic networks. For instance, a business developer maintains a track of all his contacts, meetings, bids, competitors. He currenlty has a collection of more than ten thousand subjects, split into a dozen categories. He also maintains collections about his roadrunner activity, and on building his new home.

--specialised temporary personal information management
Consultants have used Ideliance to perform missions (like assistance to tenders, audit) and to deliver their results to their customers. Instead of receiving classical static Word + Excel + Powerpoint documents, their customers can dynamically browse, query and draw graphs of the current state of the consultant work.

--knowledge management in small groups
Merck Europe Phamaceutical Labs have used Ideliance to formalize and develop a multidomain encyclopaedia on their technical, research, marketing and industrial property . They designed themselves the underlying ontology, using the semantic editor. It really changed their way of interworking among departments.

--online critical information sharing
L'Oreal, Air Liquide,Thales used Ideliance to formalize and collect information on competition. The design, formalization and collection of such information is mainly an individual task, while consultation is collective. This is a dramatic change as compared to trimestrial static documents about competition.

-- publishing the contents of classical databases in Semantic Networks
At l'Oreal, classical patents databases are translated nightly into semantic networks. The Semantic Networks facilities for browsing and querying offer, at zero development cost, a much more sophisticated interface than the original database.

-- Intelligence Applications
Ideliance is currently in operational use by French Army in France and abroad.
Since 2004, we chair the the NATO IST-38 Research Task Group on Information Fusion Demonstration, where Ideliance is one of the instrumental tools. This group is conducting an extensive study of the interest of semantic representation for Intelligence Analysts.

All these applications share the following property : users are pleased to develop critical, high value information systems themselves without the cost and lack of flexibility of traditional IT applications.

USERS ACCEPTANCE

One of our first discoveries was that users were very sensitive to the vocabulary brought by the tool. In the first versions, inspired by computer science and philosophy terminology, we used terms like « entities » « characteristics » « models ». This was strongly rejected by the targeted customers (chemist engineers, biologists, officers, medicine doctors, sales persons …). We made many evolutions, and the current vocabulary uses : Subject, Relation, Category, Statement, Collection. Yet we may have to go some steps further, down to Subject, Verb, Object, Sentence …. Needless to say, the term « ontology » is a repellent !

In the Business and Industry world, « going semantic » would be a revolution. People use the web, email and MS Office tools, and they are lightyears from the idea of leaving their daily universe of documents.
The good news is that there is a small proportion of individuals who immediately adhere to this « new way of writing ». Such people, as individual or small groups, can switch very rapidly -within days- to it, and they very seldom abandon it. Any marketing policy should leverage on these early adopters (and often true addicts !).

But the huge majority of people first rejects this new approach. To tackle the challenge of users acceptance, we propose to consider two « extremist » and opposed strategies : « Extreme Explicit Semantics » and « Extreme Implicit Semantics ».

The *« Extreme Explicit Semantics »* puts forward the idea that people should write information *directly in semantic networks*, as a substitute to producing Word, Powerpoint, Excel documents, even emails ….
The advantage of this radical approach is to set very high constraints and challenges for the design of extremely user-friendly editors. A strong consequence is also that this new kind of writing should probably be considered as a new cultural asset. The usual idea that « computerized tools should need no training, be understood and used within 5 minutes, unless they are rejected » is here completely out of the scope. People spend months and years to learn arithmetics, grammar, foreign languages, so will be the case for the semantic writing.
We experienced that a minimal course for Ideliance users lasts two days. And an ideal semantic project building session for a group lasts one full week. (of course Masters students need only 15 minutes to get fully acquainted with the tool, but have no idea or motivation for its usage).

We also felt the need to develop a methodology to teach people what semantic networks are, with a one day session *without computers* but with post-it, blackboard , games of language, and hand-written exercices. We experienced that the time lost in this day was rapidly saved later.
Indeed, in this domain, a special attention must be given to the relationship between Semantic Desktops and all the accumulated folklore, experience and tools in the domain of MindMapping.

The «*Extreme Implicit Semantics* » approach takes the opposite direction : we leave users to their favourite Office tools, and smart programs analyse the content of these documents, understand them, and implicitly, automatically build the corresponding semantic network …Whereas the *Explicit Utopia* puts all the effort on the user, the *Implicit Utopia* challenges computer scientists with the Natural Language Understanding problem. With more realism, this challenge is today approached by some text mining and automatic tagging tools.

Not surprisingly, we consider that, for future Semantic Desktops, these two approaches, Explicit and Implicit, should be accomodated. We have good reasons to hope that they will cross-fertilize : text understanding will be much easier in presence of pertinent contextual semantic resources, and the production of semantic networks will be quite accelerated by linguistic tools capable of finding or suggesting subjects and relations from texts.

Inside Thales, we have started to experiment several prototypes of bidirectional tight or loose connexion between text mining and semantic networks.

We consider that this cooperation of explicit and implicit semantic approaches is a key technology for the future Semantic Desktops.

Note : during the years of development of Ideliance, we implemented a lot of advanced buit-in features (logic programming, constraints programming, sophisticated graphs analysis algorithms). Although these features were very exciting to develop, we experienced that it was difficult for normal users to adopt them.

The first challenge Semantic Desktops will have to face is to make « semantic writing and reading » popular, before starting to complexify it. And the first step is education. Is is not a « Click and Forget » application.

IMPLEMENTATION

Since 1993, we had the opportunity to experiment several alternatives for the implementation of a Semantic Network Manager. The corresponding engine has to support strong constraints : the structure of information is more dynamic than for relational databases, algorithms –on graphs for instance- are more demanding than just sorting, joins and set theory ones. Other features like renaming subjects, fusion of collections, implementation of emergence, learning and discovery features, are uncommon.

From the lessons we learnt, our recommendations for the implementation of Semantic Desktops are the following :

--put data very close to the screen of the user - in the main memory of the desktop - and design the implementation accordingly. If you also need a light client / server implementation, upload the heavy client design to the server rather than the opposite
--consider that in the future your design will have to accomodate semantic networks filling hundreds of gigabytes of fast, random access solid state memory
--low level data structures should be as simple –even naïve- and regular as possible.
--consider persistency (on disk) as a secondary problem, to be solved separately
--the physical information representation should be fine tuned, and mastered down to the bit level. Do not trust others –like Java or Relational Databases- to take care of the rock-bottom performance.
--said differently : Semantic Networks are at the same level as both Object Oriented Languages and Relational Databases. In the same way Java is not implemented above Java and Oracle not implemented above Oracle, do not implement Semantic Networks –if you want to handle future large networks in real life - above Java and Oracle. In the case you should anyway use such high level tools, restrain yourself to their simplest features.
-- the physics of XML-based syntactical formats (or RDF, OWL, …) should not be taken into consideration for internal implementation. Their interest is in standard external exchage formats.

Semantic Desktop Computing must be considered as a very innovative, -some may say revolutionary- way to build and use information systems. In this respect, it demands and deserves implementation techniques of its own.

ROLE VISA VIS A VIS THE SEMANTIC WEB

One can play many games with the three words : SEMANTIC / WEB / DESKTOP, and « Semantic Desktop » is just one of the games. We can also consider the following ones with interest:

(SEMANTIC WEB) DESKTOP
SEMANTIC (WEB DESKTOP)
(SEMANTIC DESKTOPS) WEB
etc …

It strikes us that some leading Semantic Desktop projects like Haystack, Gnowsis and IRIS also find their motivation in personal information management, before looking at the global (semantic) web resources, although they borrow from it the basic semantic network (« triple ») representation scheme.
In this respect, we can say that the Semantic Desktops future could be absolutely independant from the existence, strength and prosperity of the Semantic Web. (After all, semantic nets –as for them- were invented by Aristotles). Ideliance illustrates this point of view, since it started even before the Web had come to a visible existence.

It is also striking to realize that the motivation for the Semantic Web was to let machines exchange information globally, whereas the motivations for the Semantic Desktop is to let humans organize information locally!

Should we consider that a P2P protocol between Semantic Desktops is still part of the Semantic Web ?

However the synergies between Semantic Personal Tools and the Semantic Web are numerous :

-- a Semantic Desktop should include a pure « Semantic Web Browser » : as « read only » as Web Browsers are
-- all the ongoing work on sharing, aligning, updating ontologies on the Semantic Web may provide useful guidelines at the smaller scale of the Semantic Desktop
-- of course, semantic nets produced by Semantic Desktops will feed the Semantic Web –provided that efficient policies to map them with Web Ontologies are designed
-- on the other hand, Ontologies and RDF producers in the Semantic Web will be motivated by all these new « clients », and also care more about the quality and usability of their data

CONCLUSION : TOWARDS « Intelligence Amplifiers »

There are many other aspects in our experience with Ideliance which we could consider in the light of the Semantic Desktop paradigm : notions of collections fusion and extraction, labelling triples with access rights, embedding text search facilities …
The main lesson we would like to transmit is that ***Semantic Desktop Computing is about people, more than about machines, architecture and protocols***. Users acceptance is key, and we badly need projects which, with the help of human and social sciences, study the cooperation between humans and Semantic Desktops.
At the end, Semantic Desktops will become « Intelligence Amplifiers »

REFERENCES ABOUT IDELIANCE

Volle Michel. Idéliance, ou "comment se faire aider par l'ordinateur pour réfléchir", in
http://www.volle.com/opinion/ideliance.htm
(A MIS consultant reports on his discovery and usage of Ideliance as a Personal Information Management Tool)

Rohmer Jean. Représentation, Fusion et Analyse d'Informations mises sous forme de Réseaux Sémantiques : vers le « Calcul Littéraire", *in « Les systèmes d'information élaborée » Société Française de Bibliométrie Appliquée, Ile Rousse, France 14-18 Octobre 2002*

Rohmer Jean. The Case for Using Semantic Nets as a Convergence Format for Symbolic Information Fusion *in NATO RTO-MP-IST-040 Information Systems Technology Panel (IST) symposium on « Military Data and Information Fusion », Prague, Czech Republic, 20-22 Oct 2003.*

Bruneau Jean-Maurice. Ideliance, Logiciel de Rupture pour l'Intelligence Economique? Un Cas d'Application sur les Signaux Faibles, *in VSST(Veille Scientifique Stratégique et Economique) 2001 Conference Barcelona, Spain*

Nabu – A Semantic Archive for XMPP Instant Messaging

Frank Osterfeld, Malte Kiesel, Sven Schwarz
DFKI GmbH - Knowledge Management Dept.
Erwin-Schrödinger-Straße, Bldg. 57
D-67663 Kaiserslautern, Germany
{frank.osterfeld, malte.kiesel, sven.schwarz}@dfki.de

Abstract

Instant messaging (IM) has become more and more common these days, and is complementing e-mail and other means of electronic communication. However, due to its heavily context-dependent nature, searching archives of instant messages using only full text search is a tedious task. Also, in contrast to mails, files, and other electronic media, instant messages typically do not feature a unique identifier or location, making it difficult to reference a particular instant messaging conversation. Nabu is a semantic archive for XMPP instant messaging designed to address these problems by implementing a semantic message store, using RDF(S) as its storage format. It is implemented as a server module and will log messages, manage access control to the archives on a per-user basis, and allow other components to observe and annotate messages.

1 Introduction

The importance of instant messaging (IM) for private and organizational communication has increased over the last years. IM, the instant sending and receiving of (mostly short) text messages between two or more users, complemented by a list of peer contacts along with their online status, has become one of the most used communication channels on the internet, and more and more valuable information is exchanged via instant messages, especially among colleagues at work.

Despite of the increasing amount of information exchanged, IM client support for archiving and searching the messages exchanged is poor. This is understandable, as on the one hand, most IM client applications are intended for private users for whom other features are more important. On the other hand, IM messages are typically very short and heavily tied to their particular context, thus making efforts to organize the archive of exchanged messages a lot more difficult than it is the case with other means of communication, such as e-mails, where the text is essentially self–contained. Moreover, e-mails come with a variety of additional information such as a subject or thread references which are usually missing in instant messages.

While e-mail can be archived in a long-term manner on server-side using the IMAP standard, there is no standard for archiving IM conversations. Chat logs are mostly stored locally on the client machine, using proprietary file formats. This has several disadvantages: storing the archive locally on the client computers is inconvenient when using more than one computer, archives are spread over different installations, and they quickly become out of sync. In addition, information gets lost easily. Using proprietary, client- and protocol-specific formats to store the information complicates managing and searching the stored information using other interfaces than the client UI.

In this paper we present Nabu[1], an open–source system providing server-side logging of instant messages. Nabu is implemented for the XML-based Jabber/XMPP protocol[2]. Unlike other proprietary IM protocols from major providers such as Yahoo!, MSN or AOL, Jabber/XMPP is an open standard. Most server and client software is available under open source licenses, which makes it possible to add Nabu's features as a plugin for an existing server implementation. The Jive Messenger XMPP server[3] was chosen due to its well-designed and well-documented code base and easy extensibility.

Nabu tries to integrate instant messaging into the efforts made in the Semantic Web [Berners-Lee *et al.*, 2001] community to store and retrieve information in a unified way. It uses the Semantic Web standard RDF[4] to describe the stored information on the server. For retrieving the stored information, it supports the SPARQL query language [Eric Prud'hommeaux, 2005], which is currently going through the standardization process at the W3C.

Using XMPP as transport protocol for SPARQL queries and commands has several benefits. For example, XMPP takes care of authentication and encryption; also, XMPP uses a persistent connection, delivering higher performance than protocols that use non-persistent connections such as HTTP, which is used as transport protocol by XML–RPC and SOAP.

In addition to the logging of chat messages, further features of Nabu are:

[1] http://nabu.opendfki.de/
[2] http://www.jabber.org/
[3] http://www.jivesoftware.org/messenger/
[4] http://www.w3.org/RDF/

- Users can add further metadata to the logged messages by adding their own RDF statements. That way information can be categorized and structured, making retrieval of relevant information easier.
- Nabu supports sharing of logged messages between users, e.g., making a conference log available to the other group members. Privacy is ensured by a strict privacy model, restricting access to explicitely authorized users.
- Nabu integrates instant messaging into the context elicitation framework of EPOS [Schwarz, 2005], sending message notifications to the EPOS user observation (when enabled by the user). Other applications can also receive these events by registering with the Nabu component.

The rest of this paper discusses related work, Nabu's architecture, the RDF schemes used, RDF access control mechanisms, Nabu's observation feature, and possible applications, followed by conclusions.

Related Work

[Karneges and Paterson, 2004] proposed a storage format and protocol for server-side message archives as a Jabber protocol enhancement. The proposal suggests a simple protocol and storage format for message archiving. It defines its own format and does not use existing standards for message storage and retrieval apart from XML.

The *Haystack project* [Dennis Quan and Karger, 2003] builds a client for information management by integrating various information sources into one frontend, using an infrastructure based on RDF. A messaging model was developed [Quan *et al.*, 2003] to represent conversations from various communication channels, such as e-mail, news groups and instant messaging, in a unified way. In contrast to Nabu, Haystack is client-based.

The *BuddySpace* research project [Eisenstadt and Dzbor, 2002] extends the *presence* concept in Jabber (simple offline/online/busy states), and adds information such as geographical location, current work focus etc. Furthermore, it investigates how such additional semantics can be used to facilitate collaboration over networks. The BuddySpace Jabber client[5] demonstrates the concepts.

2 Architecture

Nabu is implemented as a plugin for the Jive Messenger XMPP server[6]. By implementing Nabu as a server–side component, every user of the Nabu–enabled server can use its services without requiring the installation of a client–side plugin. Since there are dozens of XMPP clients[7], this was clearly the optimal solution. Also, sharing annotations would be much more difficult with a client–side implementation since clients cannot be expected to be online at all times. Finally, moving complexity to the server side nicely fits into the XMPP philosophy.

[5] http://buddyspace.sourceforge.net/

[6] http://www.jivesoftware.com/

[7] In our department, at least four different clients are in use.

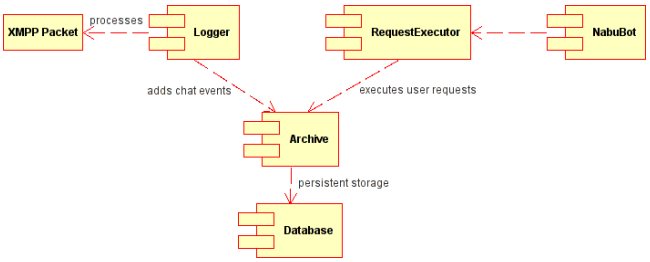


Figure 1: Nabu Components.

The graph shown in Figure 1 describes the top-level components of Nabu.

The central component of Nabu is the *Archive*. The *Archive* contains the RDF model, consisting of the public model that stores the conversation logs which can be accessed from outside, and an internal model, managing internal configuration data and privacy policies. The Nabu implementation uses Jena [Andy Seaborne et al., 2005] for RDF handling. Models are stored persistently in a database. The database backend is fully encapsulated by Jena.

The *Archive* is accessed in two ways:

- Logging: the plugin intercepts XMPP messages, converts them to RDF and stores them in *Archive*. This is done by the *Logger* component. The logger component simply takes the message, checks whether logging is enabled and if so, adds the RDF message to the RDF graph. Message URIs are created using the address of the XMPP server, ensuring uniqueness.
- User requests: the *RequestExecutor* interface allows the user to, for example, search the RDF graph. It takes parsed requests in the form of request objects, executes them, and returns a response object.

The *Nabu Bot* component is the interface between the users and the plugin. It parses user requests, creates request objects and passes them to the *RequestExecutor*. It takes the returned responses, encodes them in a string and sends them back to the requestors.

The Nabu Bot uses the Nabu protocol for transferring user queries and answers. The commands of the Nabu protocol are encapsulated as the body of chat messages. Other bindings to the XMPP protocol are possible[8] – also implementation of such a binding is quite straightforward.

In the following section, we take a look at the RDF schemas used by Nabu.

3 The Nabu Ontology

This section explains the most important parts of the ontology, i.e., the RDF schema, Nabu uses for logging. It covers the most important classes and properties for representing messages, accounts, and presence changes in RDF. Nabu handles two types of data:

- The actual RDF data that can be queried externally, i.e. logged messages, presence changes and annotations.

[8] For example, Dan Brickley's foaftown proposes an XMPP binding for SPARQL.

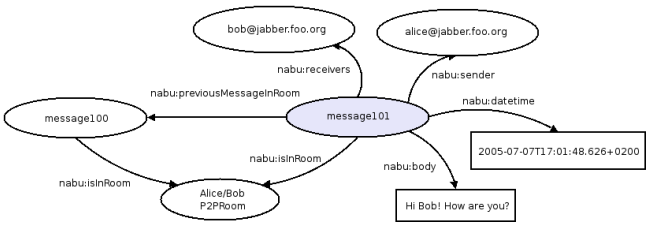


Figure 2: RDF representing an Instant Message.

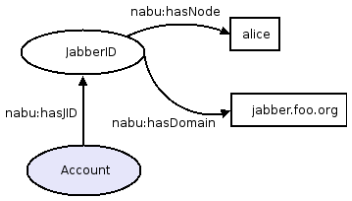


Figure 3: RDF representing an Account.

- Internal data, like account settings (e.g., logging enabled/disabled), and privacy policies. This data cannot be queried from the outside, and the users will never see the RDF representations. It is only indirectly accessible through the requests defined in the Nabu protocol.

In the following sections, the RDF schema of the data that can be queried externally will be discussed. The schemas of the data that cannot be directly queried will be presented in section 5.

3.1 Message

The most important class in the Nabu ontology is *Message*, shown in Figure 2.

- *nabu:body* is a literal containing the message text.
- *nabu:datetime* is the time stamp added by Nabu when logging the message. The format is xsd:datetime[9]. Note that if the sender and receiver are on different servers and each server has Nabu installed the time stamp will be different, so identical messages cannot be matched using the timestamp.
- *nabu:sender*: Links to the account that sent the message.
- *nabu:receivers*: The accounts that received the message. In an one-to-one chat, this is a single account, the chat partner. In multi user chat (MUC), these are all accounts that received the message, i.e. the accounts that were in the MUC room when the message was sent. Note that the temporary nick names users have in a MUC room are ignored; anonymous MUC rooms are not supported. Nick names are resolved to the corresponding accounts. Also note that the resource part of the participant's Jabber ID is omitted in both one-to-one and MUC logs.
- *nabu:inRoom*: The room the message was sent in. For details how rooms are defined see below.
- *nabu:previousMessageInRoom*: Links to the previous message in the room. This is useful for tracking conversations and for exploring logged conversations with a specific chat partner over time.

Unfortunately, in multi-user chat rooms it is difficult to track what message(s) a user is replying to. In practice, most users prefix their messages with a string denoting the receiver in chatrooms (e.g., "Frank: Please refrain from doing this."). However, Nabu does not address this issue, as other components can add annotation as needed using an heuristic.

File transfers and other activities such as video or voice chat are currently not supported. Extending Nabu to implement this functionality and subclassing the Message class accordingly should be trivial.

3.2 Accounts

The *Account* class represents a user account, as shown in Figure 3. Every user account is uniquely represented by a *Jabber ID*, like "alice@jabber.foo.org". For privacy reasons, Nabu does not store person–account associations. If such a mapping is required, one may store such information using FOAF[10], which already includes a *foaf:jabberID* property which allows linking a FOAF:Person to a Jabber ID.

Every account has a Jabber ID representing the account. However, not every Jabber ID represents an account (see MUC rooms), so Jabber IDs and accounts are not identical.

3.3 Rooms

A room is a virtual place where two or more users meet and chat with each other. Every message has one room associated, and messages in a room are linked to make conversation tracking easier. Two types of rooms exist, depending on the chat type:

In one-to-one chats, the room is defined by the two persons chatting: If Alice chats with Bob, all messages sent by Alice to Bob and vice-versa are in the "Alice-Bob-Room". The *nabu:previousMessageInRoom* property links all messages sent between Alice and Bob, making it easy for Alice to navigate through all logged messages she sent to or received from Bob. In the Nabu ontology, this kind of room is called *P2PRoom* (*Point-to-Point-Room*). In RDF, the "Alice-Bob-Room" might look like this:

```
<P2PRoom rdf:about="&foo;P2PRoom-alice-bob">
  <members rdf:resource="&foo;Account-alice"/>
  <members rdf:resource="&foo;Account-bob"/>
</P2PRoom>
```

In multi-user chat, the semantics of a room are slightly different. While the P2PRoom is a Nabu concept and does not exist in Jabber, the MUC protocol as defined in JEP-0045 [Saint-Andre, 2002] introduces the concept of rooms. A room has its own Jabber ID, just like accounts, e.g., support@conf.foo.org, so unlike P2PRooms, these "MUC-Rooms" are not defined by their members, but by the room name and topic.

[9] http://www.w3.org/TR/xmlschema-2/datatypes.html#dateTime

[10] http://www.foaf-project.org/

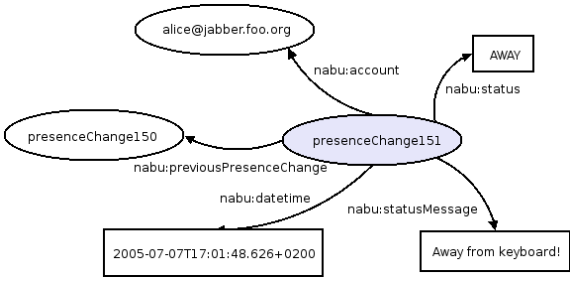


Figure 4: RDF representing Online Presence.

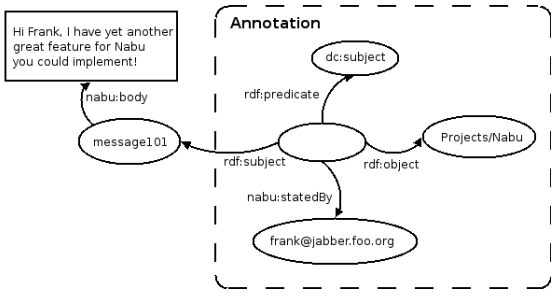


Figure 5: RDF representing an Annotation.

```
<MUCRoom rdf:about="&foo;MUCRoom-foobar">
  <hasJID>
    <JabberID rdf:about="&foo;JabberID-foobar">
       <hasResource/>
       <hasNode>support</hasNode>
       <rdfs:label>support@conf.foo.org</rdfs:label>
       <hasDomain>conf.foo.org</hasDomain>
     </JabberID>
  </hasJID>
</MUCRoom>
```

3.4 Presence Change

If presence logging is enabled, every presence change, e.g., from *Offline* to *Online* or from *Online* to *Away*, is stored in a PresenceChange instance, as shown in Figure 4.

- *nabu:status*: The new presence status.
- *nabu:statusMessage*: The status message the user set.
- *nabu:account*: The account that changed its presence status.
- *nabu:previousPresenceChange*: The last logged presence change of the account *nabu:account*. All logged presences of a user are chronologically linked via the *nabu:previousPresenceChange* property.

4 Annotations

Nabu enables users to add their own statements to the RDF store. This makes it possible for users to add metadata to logged messages and share this metadata with their peers. For example, a user could set up a set of categories to file his conversations to facilitate later searching. He could do this manually or could use a text classifier and categorize automatically. Since instant messages are heavily dependent on their context (for example, imagine a user receiving an e-mail with a question and answering via instant messaging), one may also decide to use annotations to link the messages to their context. We will discuss this in sections 7 and 8.

The user can add any statement he likes (except for statements from the Nabu schema, see below), but it is a good practice to reuse commonly used ontologies. Widespread vocabularies for metadata and categorization are Dublin Core[11] and SKOS[12]. We have to stress that Nabu does not restrict the user to annotating messages with concepts – it is perfectly possible to link arbitrary RDF constructs to chat messages. This way Nabu is flexible enough to allow tagging messages with context information such as “The user was looking at the DFKI website when sending this message”. Using Nabu’s observation features (see section 7), software running on the user’s machine can automatically annotate new messages when they arrive with information the annotation software has access to. Also, it is possible to create semantic links between messages: for example, it is possible to implement a more complex heuristic for determining the reply–chain of a message (see section 3.1 for an explanation on why this is not trivial). Adding this information to messages does not require extending Nabu – one can also write a client–side component that uses message annotations for adding this information instead.

As an example of a simple manual annotation, let us classify a (fictional) message: ”Hi Frank, I have yet another great feature for Nabu you could implement“. To make searching easier, we want to specify the project the message is related to, in this case Nabu.

The CREATESTATEMENT request adds annotations to a message:

```
CREATESTATEMENT RESOURCE
  http://&foo;Message-101
  http://purl.org/dc/terms/subject
  http://foo/Categories/Projects/Nabu
```

The first argument must be one of RESOURCE or LITERAL and indicates whether the object of the statement should be handled as resource URI or as literal string. The following tokens are the (subject, predicate, object) triple representing the statement shown in Figure 5.

The annotations are *reified*, which means that each statement itself becomes a resource that is linked to subject and object. That makes it possible to add properties to the statement. In Nabu, every user-added statement has a property *statedBy*, linking the creator of the statement. In our example, this is frank@jabber.foo.org. The statement is ”owned“ by the linked account. Only this account can delete the statement. Also, users can read the *statedBy* property and decide whether they trust the statement or not. Alice might decide that annotations made by Charlie are useful and take them into consideration, but ignore Bob’s statements.

Nearly every kind of RDF statement can be added. The only restriction is that properties from the Nabu ontology are

[11] http://dublincore.org/

[12] http://www.w3.org/2004/02/skos/

not allowed for user statements. E.g., *dc:subject* (dc = Dublin Core[13]) is valid, but *nabu:isInRoom* is not, because the predicate *nabu:isInRoom* is part of the Nabu ontology. This prevents users from currupting (deliberately or not) the Nabu archive or compromising privacy settings. Properties from the Nabu ontology are managed by the server and can only be modified indirectly by commands of the Nabu protocol.

5 Log Sharing and Privacy Settings

To gain acceptance for Nabu and server-side logging in general, it is important to ensure the user's privacy. This means that Nabu must

1. Leave the user in full control over what is logged.
2. Allow users to delete sensitive information at any time.
3. Allow access to the archive only through a clearly defined interface that handles authentication and respects the privacy settings.
4. Implement conservative default settings (i.e., disable logging, use restrictive privacy settings)

On the other hand, one of Nabu's goals is to encourage sharing between peers to make valuable information available to others when wanted. Therefore a privacy model is needed that supports both ensuring privacy and allows sharing of conversation logs.

In Nabu, every user is the owner of the messages he has sent, and he can control who can read his messages or delete them later if he wants. This means that a message is under control of the message sender only. If two users have a conversation, each user is responsible for his own messages and has no control over the messages he received from his dialog partner.

Access control is managed via *privacy policies*. A privacy policy contains a set of rules that control read permissions by allowing or denying access to certain accounts or groups of accounts. Every message logged has a link to a privacy policy that controls the access to the message, and every user has a list of policies he can assign to logged messages. There is always exactly one policy active at any one time. Whenever the user writes a message, Nabu logs the message and links it to the currently active policy. Policies are linked, not copied: For instance, if the user adds a new account to his policy "friendsOnly", the added account gains access to all archived messages that already use the "friendsOnly" policy.

What does it actually mean that a resource is not accessible? If a message (or any resource in general) is not accessible, this means the resource itself and its concise bounded description[14] is completely hidden from the user: The resource itself and all links to or from the resource are hidden. When querying the model, the resource does not show up in the results. For messages this means that neither the message content nor any links to the message are visible. This includes annotations: If an annotation was added to link the message to a category, this statement is not visible. This is important, because we do not want other users to read the topics we were talking about, even if they cannot read the actual message content.

5.1 Privacy Policies in detail

Every privacy policy has

- a name
- an owner
- a set of rules allowing or denying access to an account or a group of accounts

The name is an arbitrary string without spaces, e.g., *default*, *friends*, *workGroup*. In the requests for policy management the name is used to identify the policy. Thus the policy name must be unique for a user (but of course two users can use the same name without conflicts).

The owner is the account that owns the policy. The owner can edit the policy and add or removes rules. The policy always implicitely grants access to the owner, so the owner can access his own messages even if the rules would deny it. It is only possible for a user to change the policy for a message when he owns the currently linked policy.

The rules: A policy can contain any number of rules of the form "allowAccount <accountURI>", "denyAccount <accountURI>", "allowGroup <groupName>", "denyGroup <groupName>".

The rules are applied in (deny, allow) order. If access is not explicitely allowed, it is denied. That is, a policy without any rules denies all accesses (except to the policy owner).

If both rules exist that allow and deny access to an account, the deny-rule takes precedence and the access is denied.

5.2 Groups

As mentioned before, access permissions can be set not only per user but also per group. A group is a plain set of accounts, set up by the user to make privacy management easier. For example, a user could set up a group "friends", and add the accounts of his friends to this group. Instead of allowing access per account, he can do a simple "allowGroup friends" and all accounts in the group gain access.

5.3 Examples

Here we present some examples demonstrating how privacy policies can be applied.

There are five accounts, Alice, Bob, Charlie, Daniel and Emily. Alice is the owner of the policies, and she created a group friends with Bob and Emily in it. Note that in the implementation, full account URIs are used, but we use Alice instead of http://foo/Accounts/jabber.foo.org/alice for clarity here.

The following policy allows access to Alice (as she is the owner), Daniel and Bob.

```
policyOwner Alice
allowAccount Daniel
allowAccount Bob
```

The following policy allows access to Alice as she is the owner, friends group, which is Bob and Emily, and Charlie. So just poor Daniel may not read the resource (nor can the rest of the world).

[13] http://dublincore.org/documents/dces/

[14] http://sw.nokia.com/uriqa/CBD.html

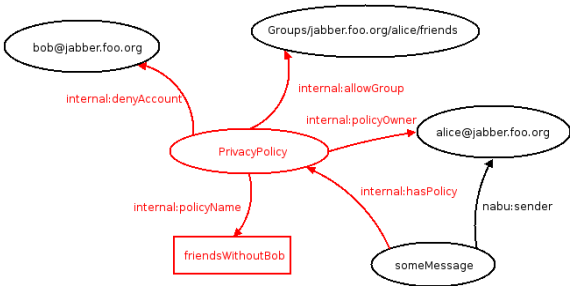


Figure 6: A Privacy Policy.

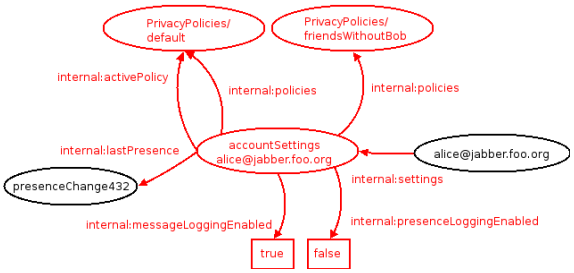


Figure 7: RDF representing Account Settings.

```
policyOwner Alice
allowGroup friends
allowAccount Charlie
```

In the following example, the first directive allows access to the *friends* group, i.e. Bob and Emily, but as the second directive denies access to Bob explicitely, only Emily has access (and Alice of course).

```
policyName friendsWithoutBob
policyOwner Alice
allowGroup friends
denyAccount Bob
```

Internally, privacy policies are realized using the the RDF shown in Figure 6. The parts of the RDF shown in black can be queried using Nabu's query features. The other parts can only be retrieved and manipulated using Nabu commands.

This graph shows the last policy example. The policy *friendsWithoutBob* is owned by Alice. It allows access to her *friends* group, but denies it for Bob. The policy is attached to a message *someMessage* which was sent by Alice. While messages, groups, and accounts are part of the public model, the privacy policy itself and all properties like allowGroup, denyAccount and hasPolicy are stored in the internal model.

5.4 Account Settings

For every account stored in the public model, the internal model contains a corresponding *AccountSettings* instance saving settings related to this account.

This example graph in Figure 7 shows the AccountSettings instance of alice@jabber.foo.org. It has the following properties:

- *internal:messageLoggingEnabled* and *internal:presenceLoggingEnabled*: Store whether message and presence logging are enabled or not. Both default to *false*.
- *internal:policies* link to the privacy policies owned by the account.
- *internal:activePolicy* links to the currently active policy. This policy is attached when a presence change or message is logged.
- *internal:lastPresence* links to the last logged presence change of the respective account. This makes it fast and easy for the logger to find the last presence and link new presences to it via the *nabu:previousPresenceChange* property.

6 Querying the Archive

Once Nabu logs a user's conversations, the user probably wants to search them at some point. For querying the archive, Nabu uses the SPARQL query language [Eric Prud'hommeaux, 2005]. SPARQL is a language for querying RDF stores, similar to SQL. Being powerful and versatile, it allows arbitrarily complex queries. Unfortunately it's also quite complex for everyday use, so a GUI for the most common queries would be desirable.

Example: One wants to search for all messages containing "Nabu". The following command performs this search:

```
QUERY SPARQL
DESCRIBE ?msg
WHERE { ?msg nabu:body ?body .
  FILTER REGEX(?body, "Nabu", "i") }
```

The query returns all messages ?msg that have a body ?body matching the regular expression "Nabu" ("i" makes the search case-insensitive). For simple string searches, there is also a shortcut available in Nabu in the form of the "QUERY SEARCHMSG" command.

Nabu will return the messages matching the query as RDF/XML. For instance, Nabu might return one message, containing "Me thinks, Nabu rocks big time":

```
210 <rdf:RDF xmlns:rdf=...
   <Message rdf:about=
     "&foo;Message-094210.520">
   <body>Me thinks, Nabu rocks big time!</body>
     <previousMessageInRoom rdf:resource=
       "&foo;Message-143802.712"/>
     <inRoom rdf:resource=
       "&foo;P2PRoom-frank2/"/>
      <subject/>
      <messageType>chat</messageType>
      <streamID/>
      <sender rdf:resource=
         "&foo;Account-frank2"/>
      <receivers rdf:resource=
         "&foo;Account-frank"/>
      <datetime>2005-07-14T...</datetime>
  </Message>
</rdf:RDF>
```

Note: Nabu returns only RDF data that has been declared as accessible. By default, this includes all messages that have been sent or received by the user who issues the query. Normally he won't see messages exchanged between other users.

If a user decides to, he can grant others access to a conversation log (for example, co-workers might decide to share the log of an online meeting with other team members).

7 User Observation

One topic addressed in the research project EPOS [Dengel *et al.*, 2002] is user observation: By observing the user's actions, EPOS tries to identify the context of the desktop in order to support the user in his work [Schwarz, 2005]. Depending on the current context of a user, different contacts, files or other resources are relevant. For instance, the context information can be used to present currently relevant contacts from the addressbook to the user. EPOS implements this using an *assistant bar*, which is a desktop panel listing relevant contacts, resources, and projects.

Collecting observation data is done by plugins for the user's applications, e.g., word processors, WWW browsers or mail clients. Each plugin observes the user's actions in the respective application and sends them to a central context elicitation component. Nabu offers this functionality for instant messaging, notifying messages from or to the observed user to EPOS.

It is important to note that in Nabu, user observation is fully controlled by the observed user. It must be activated by the user and can be stopped at any time. Observing applications need the observed password of the account in order to register at the server.

Usually observation will be integrated into the context framework by using the client API that comes with Nabu.

The observation works as follows: To observe messages from and to Alice (alice@myserver.org), the observer program logs in at the server as alice@myserver.org, like the user does with her graphical client. The observer program must use its own resource, e.g., 'observation'. To start the observation, the program sends

```
OBSERVEMESSAGES on observation
```

to the server (to test observation, this can also be sent manually to the bot). This registers the resource 'observation' as observer. From now on all messages Alice sends or receives are notified to the 'observation' resource. A notification message consists of a subject, containing the URI of the notified message, and the message body, containing the message CBD[15].

8 Applications

Numerous applications can be realized with the techniques presented. Let us enumerate some of these.

- *A message archive* – This is Nabu's most obvious application. As Nabu is a server-side component, similar to IMAP, where users may access their messages in the central archive from anywhere. Also, no inconsistencies can occur.
- *A semantic store* – As items stored in Nabu can be annotated, messages bear not only syntax but also semantics.
- *A powerful message search platform* – Using SPARQL, the archive supports both fulltext search and semantic search exploiting the relations specified in the message's annotations.
- *An exchange platform for information* – As a user's Nabu repository features fine-grained access control, other users may be granted access to a user's messages and message annotations, extending the other user's knowledge repository.
- *A gateway for integrating instant messages to your personal information model* – Nabu enables any RDF-capable software to access the user's instant messages. This way, instant messages can be integrated into the user's personal information model in frameworks such as Gnowsis [Sauermann and Schwarz, 2004].
- *A personal semantic knowledge base* – Nabu is not only about instant messages. It can store anything that may be represented in RDF. Together with Nabu access control and the access mechanisms provided by the XMPP protocol, a simple but powerful personal shareable semantic knowledge base arises. As Nabu is intended to run on a server that is continuously available, this solves problems with spurious availability of data in case the repository is implemented on the user's machine. Also, most technical problems related to reachability due to firewalls or network address translation scenarios do not occur in this approach.

9 Conclusion and Further Work

The Nabu project is an attempt to bring the Semantic Web and instant messaging together, making the increasing amount of information exchanged via instant messaging accessible using Semantic Web technology.

An ontology was developed to describe instant messaging conversations. Using the RDF standard to represent the data and the promising SPARQL query language for user queries, Nabu integrates well into existing Semantic Web infrastructures. To make better use of the stored information, users can attach metadata to their logs.

A privacy model was developed to control the accessibility of RDF data, an area where no proven implementations or standards yet exist. Working on resource-level, it is possible to control accessibility per resource. Although it has limitations when one needs more fine-grained control, like hiding only certain properties, it works well for Nabu.

The concepts were implemented as an extension for the Jive XMPP server. This proof-of-concept implementation is available[16] and can be used by interested people to integrate instant messaging and Semantic Web. In the DFKI KM working group, it is already used in the EPOS [Dengel *et al.*, 2002] project. The user observation functionality has been successfully integrated into the context elicitation.

Nabu is still a prototype. To make it suitable for widespread use, more effort and feedback is needed. The main issues are:

[15] http://sw.nokia.com/uriqa/CBD.html#definition

[16] http://nabu.opendfki.de/

- Nabu's user interface is currently text-based. This is flexible because it can be used on any platform and with any client, but is neither convenient nor user-friendly. Graphical frontends would be desirable, preferably integrated in client software (via plugins). Other options are a web frontend or integration in frameworks for desktop search.
- Evaluation is needed to find out whether the chosen privacy model meets the user requirements. This needs experience from daily use of "real users", as different usage patterns need different privacy models. At the moment, there is always one policy active at a time. The advantage is that it is easy to manage and clear which policy is used for the current chat. It would also be possible to specify a policy for specific chats, e.g., "everything I write in MUC room #workgroup should be readable by the whole workgroup". While this is more powerful, it has the disadvantage is that the user could forget about the channel-specific setting and share information with more people than intended. A third option would be to always use restrictive privacy settings when logging (i.e., only participants can read messages). Users would manually share the log afterwards by marking the conversation in their client plugin and assigning a less restrictive policy. This usage pattern is already supported, the user must just leave the default policy active, and assign other custom policies to logged messages.
- Currently Nabu is only accessible via the XMPP protocol. In order to make the repository available to software without requiring an XMPP library, it should be made possible to query the archive using HTTP(S)/XML–RPC/SOAP protocols.

Acknowledgments

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW–01 IW C01).

References

[Andy Seaborne et al., 2005] Andy Seaborne et al. Jena Semantic Web Framework, 2005.

[Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[Dengel *et al.*, 2002] Andreas Dengel, Andreas Abecker, Jan-Thies Bähr, Ansgar Bernardi, Peter Dannenmann, Ludger van Elst, Stefan Klink, Heiko Maus, Sven Schwarz, and Michael Sintek. Evolving Personal to Organizational Knowledge Spaces. Project Proposal, DFKI GmbH Kaiserslautern, 2002.

[Dennis Quan and Karger, 2003] David Huynh Dennis Quan and David R. Karger. Haystack: A platform for authoring end user semantic web applications. In *International Semantic Web Conference*, pages 738–753, 2003.

[Eisenstadt and Dzbor, 2002] Marc Eisenstadt and Martin Dzbor. BuddySpace: Enhanced Presence Management for Collaborative Learning, Working, Gaming and Beyond. Submission to JabberConf Europe 2002, 2002.

[Eric Prud'hommeaux, 2005] Andy Seaborne (edts) Eric Prud'hommeaux. Sparql query language for rdf. W3c working draft, W3C, 2005.

[Karneges and Paterson, 2004] Justin Karneges and Ian Paterson. JEP-0136: Message Archiving. Jabber Enhancement Proposal, 2004. URL `http://www.jabber.org/jeps/jep-0136.html`.

[Quan *et al.*, 2003] Dennis Quan, Karun Bakshi, and David R. Karger. A unified abstraction for messaging on the semantic web. In *WWW (Posters)*, 2003.

[Saint-Andre, 2002] Peter Saint-Andre. JEP-0045: Multi-User Chat. Jabber Enhancement Proposal, 2002. URL `http://www.jabber.org/jeps/jep-0045.html`.

[Sauermann and Schwarz, 2004] Leo Sauermann and Sven Schwarz. Introducing the gnowsis semantic desktop. In *Proceedings of the International Semantic Web Conference 2004*, 2004.

[Schwarz, 2005] Sven Schwarz. A Context Model for Personal Knowledge Management. In *Proceedings of the IJCAII'05 Workshop on Modeling and Retrieval of Context*, Edinburgh, 2005.

SAM: Semantics Aware Instant Messaging for the Networked Semantic Desktop

Thomas Franz and Steffen Staab

ISWeb, University of Koblenz-Landau, Germany
{franz,staab}@uni-koblenz.de

Abstract. While instant messaging (IM) became a mature communication means in business organizations over the last years, IM systems did not follow this evolution comparably. Communicated content is often stored insufficiently and hard to recall, integration into other desktop applications impossible. In this paper, we address these shortcomings and provide concepts for novel instant messaging. In contrast to prior work such as the Haystack system, which integrates IM data into a personal information management application, we enhance IM based on a ready to integrate ontological meta model that introduces semantics to instant messaging and its content to foster advanced management. In particular, we address networked exchange of semantic meta information to integrate IM into the Networked Semantic Desktop. The Semantics Aware Messenger (SAM) is a prototypical implementation of the concepts presented in this paper.

1 Introduction

The objective of the Semantic Desktop is to improve personal information management (PIM) by combining all content available on the desktop and relevant to the user to i) easily manage that content, regardless of which type, and to ii) simplify utilization of it.

The Networked Semantic Desktop as envisioned in [4] describes a networked infrastructure that combines the Semantic Desktop with Social Networking and P2P systems to benefit novel applications such as group collaboration.

In this paper, we address instant messaging (IM) on the Networked Semantic Desktop. In contrast to prior work such as the Haystack system [10], which integrates various desktop sources into a consistent, meta data driven personal information management application, we enhance IM based on a ready to integrate ontological meta model that introduces semantics to instant messaging and its content to foster advanced management. In particular, we address networked exchange of semantic meta information to integrate IM into the Networked Semantic Desktop.

Today, communicating by instant messaging mainly comprises typing messages and viewing incoming messages, while most of the time, no further processing of messages is done or offered so that message content gets lost in plain

text communication logs that are more or less accessible depending on the client application. Despite poor traceability, recent studies claim that IM usage has matured and IM is employed for miscellaneous tasks including complex conversation [11, 9]. Accordingly, we consider the content communicated via this media as of increasing value that should be recallable and integrated into the Networked Semantic Desktop.

The objective of this paper is to tackle traceability shortcomings of IM, improve management of IM content, and move IM towards the Networked Semantic Desktop. We are proposing i) an ontological meta model for instant messaging which ii) supports integration into the Networked Semantic Desktop, and iii) introduces meta data and semantics for IM to enable iv) sophisticated reutilization of instant messaging data. Based on the meta model and an v) identification scheme for IM data including meta information and semantics we vi) enable networked exchange of such information via IM to vii) ground novel applications as envisioned in [4].

In Sect. 2, we sketch a typical IM scenario to indicate shortcomings of current IM systems (Sect. 3), explain our concepts to overcome these shortcomings (Sect. 4), and illustrate the implementation of these concepts by examples of that scenario in Sect. 6. We give a detailed overview of the ontological meta model in Sect. 5, and contrast our work with related work to provide a conclusion in Sect. 7. In Sect. 8, we suggest future research and give an outlook.

2 Scenario

The extracts of chat conversations in this section render a common instant messaging scenario and indicate different functions and particularities of IM.

The scenario: Steffen, being the lecturer of the Semantic Web lecture uses IM to get some quick responses concerning organizational issues from Thomas, who held the last exercise session for the lecture:

```
[09:29:06] Steffen: how was the exercise session?
[09:29:33] Steffen: did you tell them the date of the exam?
[09:30:33] Thomas: solutions were ok, participation was weak
[09:30:56] Thomas: yes, i guess about 20 will sign up for it
```

Listing 1.1. Exercise Session

At a later time, Thomas informs Steffen about his work on a paper he is writing for the Semantic Desktop Workshop.

```
[12:07:45]  Thomas: i will put new versions of the paper for
the sdws at http://isweb.papers.x.y/sam.tex
[12:08:18] Steffen: ok, what is going to change?
[12:08:35] Thomas: describe an IM scenario to indicate current
shortcomings, propose improvements, and demonstrate SAM
in terms of the scenario
```

Listing 1.2. Semantic Desktop Paper

After lunch, Thomas contacts Steffen about the scenario he mentioned in Listing 1.2.

```
[13:55:04] Thomas: any ideas for a suitable scenario?
[13:55:24] Steffen: why don't use this conversation?
[13:55:43] Thomas: right! its a sufficient example of a
work related chat
[13:56:15] Thomas: i'll use our today's earlier chats
as well. they nicely indicate different functions of IM
```

Listing 1.3. Scenario for the Paper

Later on, Steffen talks to Bernhard, a co-worker in project X:

```
[17:15:42] Steffen: wrt the project you might be
interested in what thomas is currently doing; i'll send
you what thomas told me about that so far
[17:16:03] Bernhard: thanks, i'll contact him when
i've read it
```

Listing 1.4. Project X Work

2.1 Terminology and Observations

Isaacs et al. [9] discovered that – in professional environments – IM messages mostly are work-related (61.8%), followed by scheduling and coordinating ones (30.8%) and those that resemble simple questions and information (27.8%).[1]

Based on that terminology, we classify the chat excerpts (Listing 1.1 to 1.4) as follows: Listing 1.1 is a sample of *simple questions and information*, while Listings 1.2, 1.3, 1.4 represent *work-related* messages. In the given scenario, we excluded scheduling/coordinating conversations, as they resemble a typical IM function, but do not contribute much here.

2.2 Use Cases

Due to the fact that most IM conversations are about work, the content of such conversations needs to be available for later reuse as illustrated by the two following use cases.

Use Case 1: About one week after the day when the listed conversations took place, Steffen wants to check where Thomas stored that file on the server, and what exactly he stated about his current work. As Thomas is not available he cannot ask him again.

Use Case 2: In order to track project development, and summarize the current stage of project X, Steffen wants to compile all project X related content, including messages that deal with the project.

[1] Messages could be classified for more than one category.

3 Accomplishing the Use Cases Today

Today's instant messengers usually store messages in plain text logs and provide a user interface to view the logs, sometimes ordered by message date or filtered by user. More sophisticated messengers may supply an additional search over the message logs. Accomplishing the use cases with current systems reveals the following shortcomings:

1. Weak Message Classification:
   Finding appropriate messages by browsing the message logs requires high user effort as the given classifications (by user, by date) do not narrow the search space enough to easily find messages: Given that Steffen does not recall the exact day when Thomas told him about his current work, he has to read all the messages from several days to find the one he seeks.
2. Keyword search is unsuitable due to missing content semantics and particularities of chat conversation style:
   (a) A term denoting the subject of a message, or significantly distinguishing a message from others is not necessarily contained in a message so that creating efficient search strings is delicate. Entering a query that finds the message Steffen looks for in use case 1 may be difficult as Thomas did not use keywords like "store", "server", or "file" that directly relate to the semantics of his message in Listing 1.2.
   (b) Ambiguity of search terms further decreases the average relevance of search results. If Steffen searches for *paper*, he may receive messages that deal with different concepts of paper such as *writing paper*, *abrasive paper*, and *research paper* while only the latter is relevant for him.
3. Missing Context:
   (a) Instant messages are rather short, and informal [7, 6, 12] therefore become meaningless without context. In Listing 1.1, Thomas said "the solutions were ok, participation was weak". Without the message's context it is hard to predict which solutions Thomas points at. Current IM systems do not provide message context so that identifying relevant messages is difficult.
   (b) Topic switching and interleaving messages are particularities of IM conversation. Listing 1.1 has interleaving messages, as Thomas' first message replies to Steffen's first message although it appears after Steffen's second message. The context of interleaving messages is not based on the sequence in which they appear in time so that even browsing message logs ordered by time does not necessarily provide relevant context.
4. Missing Messaging Semantics:
   Current IM clients do not identify message properties, e.g. the creation date, or sender of a message. Consequently, relations between them cannot be exploited:
   (a) Missing messaging semantics inhibit integration into the Networked Semantic Desktop.
   (b) Information exchange is of low value as just meaningless plain text can be exchanged.

(c) Semantic querying using restrictions on properties is impossible, e.g. querying for messages within a date range, sent by a certain user et cetera.

4 Improvements by SAM

4.1 Message Classification for Message Semantics

The first shortcoming mentioned in Sect. 3 denotes weak message classifications provided by current IM clients. SAM offers a user-definable taxonomy that is used to add semantics to messages by annotating them with entries from the taxonomy. For instance, Steffen might define the category *work* with two subcategories *teaching* and *projectX*. If he annotates any message related to project X with the corresponding entry in the taxonomy, accomplishing the second use case is as easy as browsing for all messages annotated with *projectX*. Message classification also benefits search, as queries can restrict search results to be annotated with certain taxonomy entries. How annotations and the taxonomy are designed is detailed in Sect. 5, how the user annotates with SAM is explained in Sect. 6.2.

The main drawback of message classification is the user effort required to annotate messages appropriately. This effort is lowered by automatic annotation exchange between conversation partners as detailed in Sect. 4.3 and 6.4, however, manual annotation still has to be done by at least one of a conversation's participants in order to gain benefits. The user interface of SAM tries to minimize this effort as much as possible (see Sect. 6.2) and for future work we propose to integrate automatic message classification based on machine learning technologies.

4.2 Ontological Meta Model

We employ a meta model for instant messaging in form of a unified messaging ontology (cf. Sect. 5) that tackles many of the shortcomings listed in Sect. 3.

The ontological meta model provides semantics for IM entities such as persons, messages, conversations, annotations, and message texts as it identifies and relates such entities to each other by meaningful properties. This permits several enhancements as detailed in the following:

Message Context: Any message is accompanied by its context, i.e. messages link to their following message, their sender and recipient and so on. Accordingly, messages displayed while browsing or in search results are much more informative thus reducing the user effort of determining whether or not they are relevant.

Semantic Querying: Querying becomes more powerful as the ontological meta model permits to define what to query for, e.g. one can not only query for messages but also for users or taxonomy entries. Moreover, restrictions on properties

can be defined, e.g. Steffen can request messages sent by Thomas within a certain date range, including the keyword "paper" in their message text. Resulting messages will directly link to related messaging entities to provide context.

Integration: As the ontology unambiguously defines messaging entities it integrates IM into the Networked Semantic Desktop by providing interoperability between applications. For instance, the sender of a message in Steffen's store can be identified as the author of a document on his hard disk, or the sender of an email in his email client. Such features require, however, that applications commit to the same ontology. Thus, SAM does not employ a proprietary representation of persons, but integrates the Friend-of-a-Friend (FOAF[2]) ontology as it is widely recognized for expressing identity.

The ontology abstracts the concept of a message considering interoperability of different message channels as proposed in [13]. A unified view of messaging aims at seamless integration between different messaging applications as it allows to track conversations that comprise different message types and message channels, e.g. receiving an email message and answering with an instant message.

4.3 Meta Data Exchange

All participants of a conversation deal with the same set of messages. As each user decides how to annotate a message and which concepts to have in his taxonomy, there are cases where annotations differ between users, and where one user annotated a message while the other one did not. A common meta model on each peer, unique identification of IM entities, and provenance information established by the messaging ontology enables automatic annotation exchange between peers to either add further message semantics through additional annotations, or add annotations for not yet annotated messages. The latter case is especially important to reduce annotation effort for the user. As each user maintains his own taxonomy, annotation exchange may also introduce new taxonomy entries. SAM offers different user options to deal with incoming annotations as explained in Sect. 6.2. Technical aspects of meta data transfer are mentioned in Sect. 6.4.

Meta data exchange is not only useful to decrease annotation effort, it permits several novel applications. In Listing 1.4, Steffen tells Bernhard to send him, what Thomas told him. Meta data exchange as proposed by SAM allows to automatically integrate messages sent between Thomas and Steffen into Bernhard's data store so that Bernhard can utilize all features of SAM to access these messages.

5 The Ontology

Figure 1 depicts the ontology and defines the namespaces used for the following textual explanation of the ontology. A conversation is modeled by the class

[2] http://www.foaf-project.org/

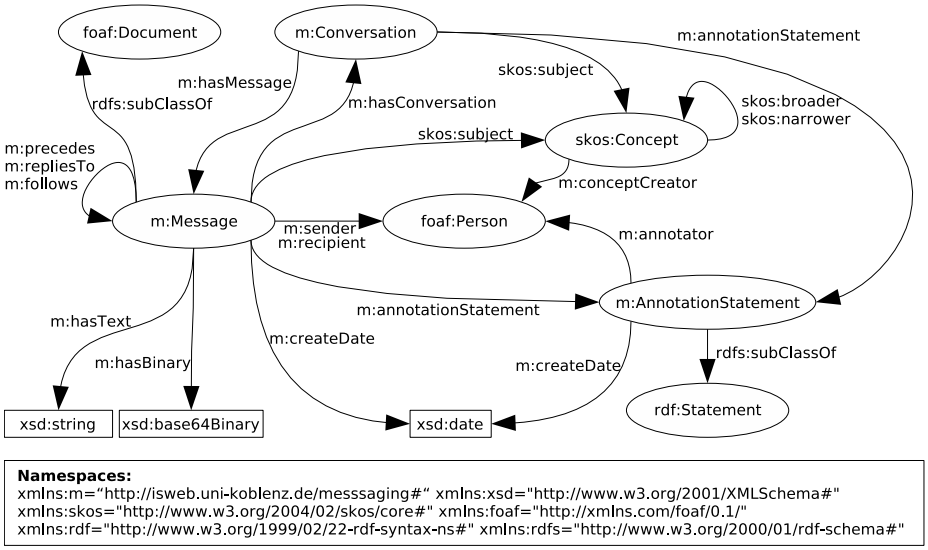


Fig. 1. Unified Messaging Ontology of SAM

`m:Conversation`, which relates to messages exchanged within a conversation by the `m:hasMessage` property. A message is a subclass of `foaf:Document` and is associated to its content by the `m:hasText` and `m:hasBinary` properties. The `m:follows` property and its inverse, `m:precedes`, track the chronological order in which messages appear, while the `m:repliesTo` property records further valuable context information that goes beyond chronological ordering: It relates a message to the message it replies to thus relating these messages based on the semantics of their content. This property is significant to store appropriate context information for interleaving messages as illustrated in Listing 1.1. Section 6.2 explains how this property is set using SAM.

Persons are represented by `foaf:human` as defined in the FOAF ontology, which already features messaging relations, including instant messaging properties such as `foaf:jabberID`.

In order to add semantics to messages and conversations, they are annotated with entries of a taxonomy. The taxonomy is defined using the Simple Knowledge Organization System (SKOS[3]), an ontology to describe concept schemes providing several predefined classes and properties for this purpose. The `skos:narrower` and `skos:broader` properties are used to build a `skos:Concept` hierarchy, while the `skos:subject` property is used to associate *things* - in our case messages and conversations - with concepts.

Employing a standard meta ontology for knowledge representation fosters integration of ontologies that are based on the same meta ontology. However, as the hierarchical structure is established by only two relations, namely *broader*

[3] http://www.w3.org/2004/02/skos/

and *narrower*, transforming existing taxonomies or lexica defined with other meta ontologies to a SKOS representation is straightforward as well. As an example, Wordnet[4] can be transformed to a concept hierarchy defined with SKOS by interpreting the hypernym and hyponym relations of Wordnet as narrower and broader relations of SKOS.

Provenance data for annotations that allows to track who annotated what and when is established by individuals of `m:AnnotionStatement` that references the creator (`m:annotator`) and creation date of an annotation. Any such annotation is a reified statement that points at the resources representing the annotation.

Provenance information is also kept for messages and taxonomy entries by the `m:sender`, and `m:conceptCreator` properties as illustrated in Fig. 1.

6 SAM

6.1 Technologies Enabling SAM

SAM builds upon the instant messaging client BuddySpace[5] [17], which was developed during research on online presence in instant messaging at Open University. BuddySpace is a client for the Jabber[6] network which we extended to use the ontology depicted in Sect. 5. A programming interface was developed that encapsulates the ontological model and provides methods to write to it and read from it, such as adding an annotation, or retrieving messages annotated with a given concept.

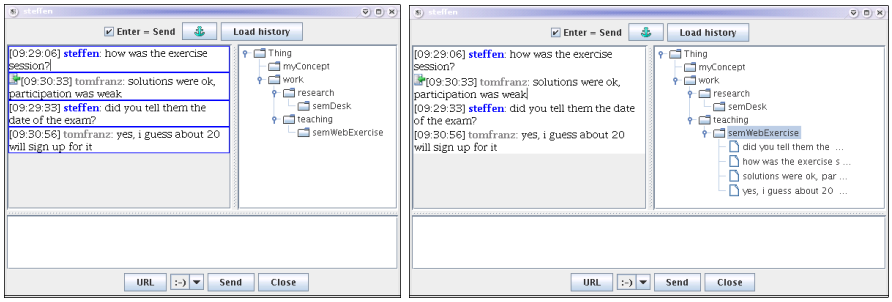
The messaging ontology is defined using the Web Ontology Language (OWL)[1]. It defines the properties and classes as explained in Sect. 5, including appropriate restrictions for them (range, domain, cardinality, functional, inverse, et cetera). Instances of the classes defined in the ontology are represented as RDF to support integration with the Networked Semantic Desktop and to establish a well structured and easy to access data store that simplifies incorporation of meta information, interlinking of resources, and exchange. The Jena[7] RDF API for Java is used to access the store.

The communication protocol used by the Jabber network is the Extensible Messaging and Presence Protocol (XMPP)[15], an XML-based protocol that is well supported by multiple open source programming libraries.

6.2 Annotations and Context

In contrast to common IM clients, the chat window of SAM contains an additional taxonomy panel (cf. Fig. 2). The chat window permits message annotation, taxonomy management, and the addition of context information while chatting. Both, the message panel and the taxonomy panel allow to accomplish multiple

[4] http://wordnet.princeton.edu
[5] http://kmi.open.ac.uk/projects/buddyspace/
[6] http://www.jabber.org
[7] http://jena.sourceforge.net/



(a) Selecting multiple messages. (b) Annotating with a taxonomy entry.

Fig. 2. Annotating Multiple Messages

annotations at once to reduce user effort. Annotations are made either by double-clicking on a particular message that automatically annotates that message with all taxonomy entries that are currently selected, or by double-clicking a taxonomy entry which automatically annotates all selected messages with that entry as illustrated in Fig. 2. To further minimize user effort, if no message is selected, double-clicking on a concept contained in the taxonomy automatically annotates the last displayed message. As direct visual feedback, annotated messages are displayed as child nodes in the taxonomy (cf. Fig. 2b).

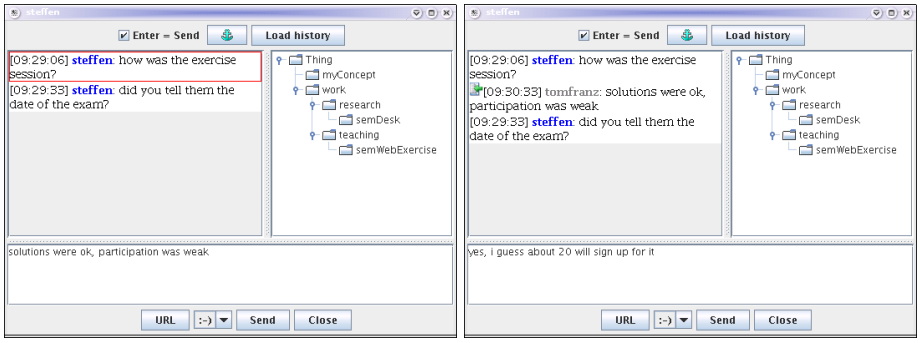
New annotations are automatically sent to the conversation partner to further reduce annotation effort and gain additional message semantics. We propose different policies (cf. Table 1) that define how new annotations that potentially introduce new taxonomy entries are handled based on how much trust is given to the creator of an incoming annotation.

Table 1. Policies for handling incoming annotations.

Trust Level	**New Annotation**	**New Taxonomy Entry**
Low	require user confirmation	require user confirmation
Medium	automatically add annotation	require user confirmation
High	automatically add annotation	automatically add entry

For any created message, the `m:follows`, `m:precedes`, `m:sender`, `m:recipient`, `m:hasText`, and `m:hasConversation` properties are automatically set by SAM to establish context information. The `m:repliesTo` property can be set through the message panel of the chat window as illustrated in Fig. 3: Selecting a message with a right-click automatically sets the `m:repliesTo` property of the next sent message to the selected one. Messages that have this property set are automatically displayed underneath the message they reply to. As IM conversations often have interleaving messages (cf. Listing 1.1) with different topics, this feature does not only provide additional message context, but also eases IM conversation as it

assists the user in identifying related messages. All context information created for a message on one client is automatically transferred to the recipient when that message is sent to provide as much meta information as possible on both sides of a conversation. Section 6.4 describes in more detail how the transfer of such information is implemented.



(a) Selecting message to reply to. (b) After sending the message.

Fig. 3. Replying with interleaving messages

6.3 Semantic Search and Semantic Browsing

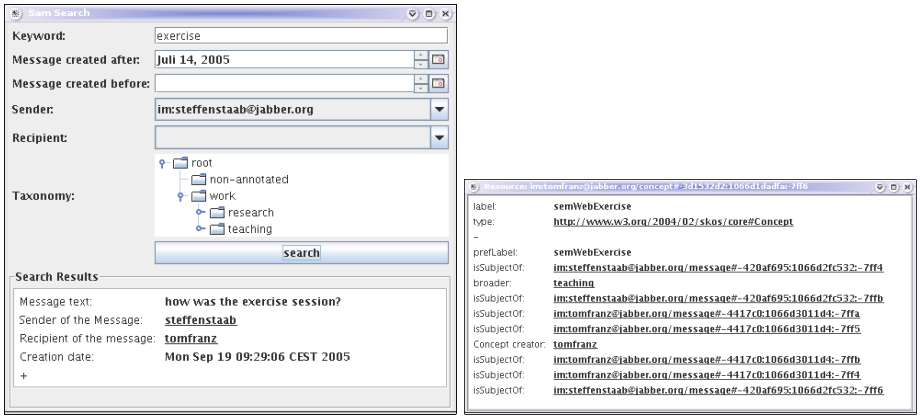
SAM allows to combine full-text search in message texts with semantic search features as illustrated in Fig. 4a. The user can restrict a search by specifying a date range for the message creation time, require specific persons to be the sender and the recipient, and restrict search results to be associated with certain taxonomy entries. Resulting messages are displayed with their context available for further exploration through the property explorer that opens by clicking on non-literal objects such as persons and taxonomy entries (cf. Fig. 4b).

The semantic browser (cf. Fig. 5) allows to view messages classified by the individual taxonomy. Non-annotated messages are associated with an additional taxonomy entry so that the user can still access them. As for search results, object properties (displayed underlined) can be further examined (cf. Fig. 4b).

6.4 Meta Data Transfer

Every messaging entity (e.g. person, message) is identified by its uniform resource identifier (URI) to support global identification and thus exchange of such entities. For example, each new instance of `m:Message` needs to be available for the sender and the recipient as both may want to reutilize it.

SAM exploits the extension mechanism of the XMPP to transfer messaging entities between different SAM clients, and to support automatic meta data exchange. Different extension types, namely *message*, *annotation*, *resourceRequest*,



(a) Semantic search. (b) Property Explorer.

Fig. 4. Semantic Search and Property Explorer

and *resourceResponse* are defined for this purpose. A *message* extension contains the RDF representation of a message while an *annotation* extension contains an instance of a `m:AnnotationStatement`. The two other extensions enable to request and retrieve one or multiple RDF resources with all their properties. The following two use cases exemplify how the extensions are used:

1. When sending a chat message, SAM automatically creates a new instance of `m:Message` with corresponding properties, and attaches its RDF representation in a *message* extension to the XMPP packet that sends the message. The receiving SAM client extracts the RDF data contained in the packet's extension and adds it to its own store.
2. When a client receives an annotation with a taxonomy entry that is not contained in its RDF store, the client repeatedly requests more general (`skos:broader`) taxonomy entries from the sender until a retrieved entry matches an entry in the local taxonomy so that the new taxonomy entry can be correctly inserted into the taxonomy and the annotation becomes effective.

7 Conclusion & Related Work

This paper presents concepts and an implementation of enhanced IM with respect to the Social/Networked Semantic Desktop. SAM introduces rich meta data, including semantics, to instant messaging and its content to provide enhanced management features that exploit such additional information. The main achievement, distinguishing SAM from existing systems, is the establishment of an IM infrastructure to globally exchange content and its semantic meta data in order to gain knowledge. This ability grounds several novel applications such as knowledge collaboration.

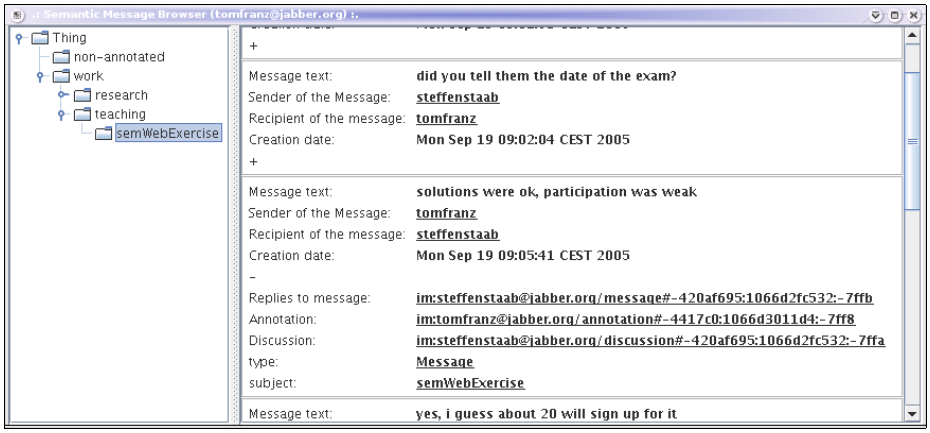


Fig. 5. The Semantic Browser of SAM

Zhang et al. present the *Small World Instant Messenger* in [18]. They build user profiles based on users' bookmarks or homepages, which are then used for expertise search. In contrast to our approach they rather exploit the infrastructure provided by instant messaging without addressing any issues of instant messaging itself. Consequently, they disregard management, reusability, and integration issues while establishing a service on top of instant messaging.

The Haystack system [14] comes with a general notion of messaging including a unified messaging ontology [13] similar to the ontology presented in Sect. 5. While the Haystack system focuses on integrating messaging into a personal information manager, SAM considers the networked exchange of meta information and is ready to integrate with other applications on the Semantic Desktop.

Chirita et al. explain how to use activity based semantic meta data [3] in their desktop search prototype. Exemplarily, they deal with email, file system, and web cache meta data and have developed an architecture that combines such meta data with standard full text search. While our work also combines full-text search and meta data to improve management, we address different enhancements and options for exploitation that are specific to the instant messaging context.

Vogiazou et al. established enhanced symbolic presence for instant messaging [17]. One outcome of this research is the BuddySpace instant messaging client and server component that allow to automatically group *buddies* and visualize their location and presence information respectively. SAM extends BuddySpace by semantic annotations, semantic search, semantic browsing, and (semantic) meta data communication.

The CoAKTinG (Collaborative Advanced Knowledge Technologies in the Grid) project [2] developed a meeting ontology to summarize content of different collaborative technologies. The summarized content is used to provide meeting replays that span content communicated via multiple channels, such as instant messaging, or video conferencing. While CoAKTinG imports BuddySpace com-

munication logs into the meeting ontology, SAM contributes to CoAKTinG by providing already well structured additional (semantic) meta data.

The Gnowsis system [16] provides an architecture and server component for integrating arbitrary applications on the Semantic Desktop. Applications are required to describe their data by ontologies and are connected to the Gnowsis desktop service by plugins. As a result, different data from various desktop applications is unified through a single Gnowsis user interface. As SAM already employs ontologies to represent all its data, integration into the Gnowsis system is at hand.

8 Outlook

Meta data exchange as explained in Sect. 4.3 and 6.4 can enrich knowledge bases but also institutes several applications that go beyond that scope. Taxonomy overlappings between different communication partners represent a shared view, naturally established based on communication of taxonomy entries and their relations. Accordingly, rejecting and accepting incoming taxonomic data is a simplistic example of online collaboration on a concept hierarchy. Further work on generalizing the process model will allow online collaboration that is independent of a specific problem domain.

While IM is employed by business organizations, improving company wide knowledge management through expertise search might be a welcomed feature in businesses. The Bibster project [8] establishes semantic routing based on the expertise of peers. In Bibster, expertise is computed from annotations of bibliographic data with topics from the ACM topic hierarchy. The knowledge base provided by SAM can be exploited similarly, however, not to implement semantic routing but to compute the expertise of users and provide an expertise search.

As mentioned in Sect. 4, we consider automatic message classification as a future improvement. As instant messages differ from other text documents [9, 12], we consider classification of such messages as a challenging task. However, as SAM provides rich message context, any message is usually related to several other messages that may be exploited to improve classification. Moreover, if each SAM client runs a classifier that works on a potentially different knowledge base, we may investigate how to combine different classifiers and their results to improve overall classification quality.

A very significant open issue is how to incorporate security and privacy issues, especially trust as defined in [5] as credibility and reliability of resources.

Application oriented visions include the integration with existing software for the Semantic Desktop such as the Haystack or the Gnowsis systems.

Acknowledgments

We would like to thank Arup Malakar for his contributions to the development of SAM. This work is conducted with respect to the upcoming project Knowledge

Sharing and Reuse across Media (X-Media), funded by the Information Society Technologies (IST) programme of the 6th Framework Programme.

References

1. Grigoris Antoniou and Frank van Harmelen. Web Ontology Language: OWL. In *Handbook on Ontologies*, pages 67–92. 2004.
2. Michelle Bachler, Simon Buckingham Shum, Yun-Heh Chen-Burger, Jeff Dalton, David De Roure, Marc Eisenstadt, Jiri Komzak, Danius Michaelides, Kevin Page, Stephen Potter, Nigel Shadbolt, and Austin Tate. Collaboration in the Semantic Grid: a Basis For E-Learning. In *Grid Learning Services Workshop, 7th International Conference On Intelligent Tutoring Systems*, pages 1–12, 2004.
3. Paul-Alexandru Chirita, Rita Gavriloaie, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Activity Based Metadata For Semantic Desktop Search. In *ESWC*, pages 439–454, 2005.
4. Stefan Decker and Martin R. Frank. The Networked Semantic Desktop. In *WWW Workshop On Application Design, Development and Implementation Issues in the Semantic Web*, 2004.
5. Jennifer Golbeck, Bijan Parsia, and James A. Hendler. Trust Networks On the Semantic Web. In *CIA*, pages 238–249, 2003.
6. Rebecca E. Grinter and Margery Eldridge. Y Do Tngrs Luv 2 Txt Msg? In *ECSCW*, pages 219–238, 2001.
7. Rebecca E. Grinter and Leysia Palen. Instant Messaging in Teen Life. In *CSCW*, pages 21–30, 2002.
8. Peter Haase, Jeen Broekstra, Marc Ehrig, Maarten Menken, Peter Mika, Mariusz Olko, Michal Plechawski, Pawel Pyszlak, Björn Schnizler, Ronny Siebes, Steffen Staab, and Christoph Tempich. Bibster - A Semantics-Based Bibliographic Peer-to-Peer System. In *International Semantic Web Conference*, pages 122–136, 2004.
9. Ellen Isaacs, Alan Walendowski, Steve Whittaker, Diane J. Schiano, and Candace A. Kamm. The Character, Functions, and Styles of Instant Messaging in the Workplace. In *CSCW*, pages 11–20, 2002.
10. David R. Karger, Karun Bakshi, David Huynh, Dennis Quan, and Vineet Sinha. Haystack: A General-Purpose Information Management Tool For End Users Based On Semistructured Data. In *CIDR*, pages 13–26, 2005.
11. Michael J. Muller, Mary Elizabeth Raven, Sandra L. Kogan, David R. Millen, and Kenneth Carey. Introducing Chat Into Business Organizations: Toward an Instant Messaging Maturity Model. In *GROUP*, pages 50–57, 2003.
12. Bonnie A. Nardi, Steve Whittaker, and Erin Bradner. Interaction and Outeraction: Instant Messaging in Action. In *CSCW*, pages 79–88, 2000.
13. D. Quan, K. Bakshi, and D. Karger. A Unified Abstraction For Messaging On the Semantic Web. In *The Twelfth International World Wide Web Conference*, 2003.
14. 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.
15. Ed. Saint-Andre. RFC 3920: Extensible messaging and presence protocol (XMPP). Technical report.
16. Leo Sauermann. The Gnowsis Semantic Desktop For Information Integration. In *WM 2005: Professional Knowledge Management*, pages 39–42, 2005.

17. Yanna Vogiazou, Marc Eisenstadt, Martin Dzbor, and JiríKomzak. From Buddyspace to CitiTag: Large-Scale Symbolic Presence For Community Building and Spontaneous Play. In *ACM SAC*, pages 1600–1606, 2005.
18. Jun Zhang and Marshall W. van Alstyne. SWIM: Fostering Social Network Based Information Search. In *CHI Extended Abstracts*, page 1568, 2004.

Semantic Social Collaborative Filtering with FOAFRealm

Sebastian Ryszard Kruk, Stefan Decker

Digital Enterprise Research Institute, Galway, Ireland*
<firstname.lastname>@deri.org, http://www.deri.org

Abstract. The most popular collaborative filtering implementations require either a critical mass of referenced resources and a lot of active users. Other solutions are based on finding a referral with an expertise on the given domain of discourse.
In this article we present the semantic social collaborative filtering solution to information retrieval. We describe how the concept of users' managed collections can be exploited to provide collaborative filtering system based on social network maintained by the users themselves.
We present FOAFRealm, a user profile management system based on the social networking and the FOAF metadata. FOAFRealm enables distributed collaboration between parties in the semantic social collaborative filtering way.

1 Introduction

The contemporary Internet contains a lot of information. In the unorganised structure of the Web all the information that we are looking for seems to be always just behind the corner. Though, still beyond our scope. And when we fail to find that information, it turns to be useless. Search engines and online catalogues tend to return a lot of resources as an answer to our queries. Very often some of results are unrelated to given queries. No wonder, we end up asking our friends and acquaintances for interesting references on the exact topic. Collaborative filtering is an idea of automating the process of asking around when looking for the information on the Internet[1].

Since early implementations of collaborative filtering, like introduced in [2], a number of methods have been developed for the "collaborative filtering" and "social filtering" [2–4]

Contributions The paper makes the following contribution to the field of collaborative filtering and user profile management systems:

* This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131. Authors thank all members of the JeromeDL (cf. http://www.jeromedl.org/) and the FOAFRealm (cf. http://www.foafrealm.org/) working groups for fruitful discussions on this document.

– We introduce a new approach to collaborative filtering - the semantic social collaborative filtering that covers both active and passive types and solves additionally some privacy/security issues.
– The reference implementation library (FOAFRealm), can be embedded into of web applications, providing additionally unified, distributed users management system based on FOAF.
– Our solution introduces goals like: distributed user profile management, privacy of the profile information, security of the provided knowledge, utilisation of social networks.

Outline of the paper The next section describes the architecture of the semantic social collaborative filtering in the context of other similar solutions. In section 3 we present the evaluation of the underlying model of social interactions in the semantic social collaborative filtering. We describe in section 4, the FOAFRealm system that implements a distributed user profile management system and delivers semantic social collaborative filtering features. Later, we discuss the relations between social collaborative filtering and digital library systems.

2 Semantic Social Collaborative Filtering

The semantic social collaborative filtering presented in this article is based on two concepts: distributed collections and annotations of resources. Each user classifies only a small subset of the knowledge, based on the level of expertise he/she has on the specific topic. This knowledge is later shared across the social network.

2.1 How does Social Collaboration Work

The problem that there is a trade-off between accuracy and scalability is often found in search engine applications. The information gathered in online collections is very precise, as the human factor is involved in the indexing process. But since the Internet is growing so fast, the process of creating the catalogue does not scale. On the other hand, search engines do the indexing work without involving the human activity. And results of queries are not always satisfiable.

A social network is a set of people or group of people, with some pattern of interactions or "ties" between them[5–8]. A social network is modeled by a digraph where the nodes represent individuals, and a directed edge between nodes indicates direct relationship between two individuals.

It is possible to construct a subgraph, on top of a social network, that represents flow of expertise in the certain domain. The idea of the semantic social collaborative filtering is based on this observation. Each person in the social network gathers the interesting information in collections he/she has created. Collections maintained by other people can be easily linked into own collections created by the user. As we show later (see section 3) the information disseminated through the collections linking across the social network corresponds to the expertise level on particular subject in the social network.

Distributed collections. The information is gathered in colletions by a number of people. Each of them handles specific domains of discourse within the colletions information space he/she has created. The quality of the information gathered across the colletions can be satisfied by approving the expertise in given domain of discourse.

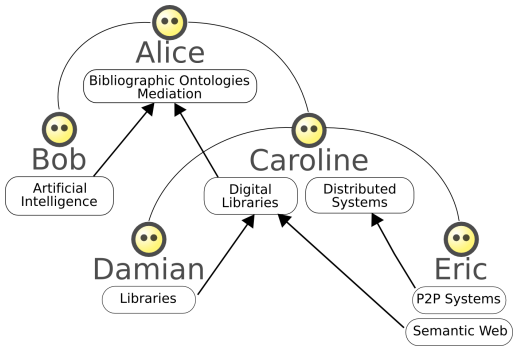


Fig. 1. The scenario of a simple semantic social collaborative filtering model

Each user maintains his own collections (private bookshelf [9]) and renders them accessible to his/her friends [10]. We can assume that some of topics are better explored by some people. Each collection has a quality level assigned to it, based on the expertise the owner has on the related topic. Each user is also aware of the expertise level of other people on given topics.

Resources annotations. Apart from managing collections by providing the categorisation description of resources, the semantic social collaborating filtering utilises comments and annotations provided by the users. The annotations are represented as fora with some additional semantic content. Annotations can be used by other people as a shorthand to quickly explore: (1) the content or meaning of the resource; (2) the context of resources; (3) the general opinion of other users.

2.2 Semantic Social Collaborative Filtering Scenario

In our example scenario, Alice writes a thesis on "Mediation in Bibliographic Ontologies". She registers to the digital library run by the University. She discovers that some of her friends are already registered to the library as well. With features based on online communities, she connects her profile to her friends profiles. Later on, Alice starts to gather the information required for her thesis topic. She keeps links to resources she has found in collections managed by the online bookmarks system. Soon she discovers that resources that she has bookmarked do not cover the topic of the thesis at satisfiable level. The following sections describe different algorithms Alice uses to find the desired information with the help of the semantic social collaborative filtering.

Simple Social Collaborative Filtering To find the desired information Alice sings up to the university digital library. The system used by the library is based on the simple semantic social collaborative filtering implementation (see Fig. 1).

Alice uses the searching features provided by the digital library web application (see Fig. 2(a)) to find interesting resources.

We introduce a solution to the problem stated in previous section (see 2.2) – a simple semantic social collaborative filtering model (see Fig. 2(b)). Each collection is categorised by the owner. Collaborative filtering feature in the digital library lists all the collections, within the given range of friendship neighbourhood, with topics related to the ones defined by Alice. Each collection has a quality level assigned to it. The quality of the collection corresponds to the expertise level of the owner on related topic. The expertise level can be computed with PageRank algorithm applied to graphs of collections inclusions and social network. Both graphs represents the rank value each person and each collection receives from other people. The rank values are assigned directly (by people to people) and indirectly (by including someone's collection to own collection).

Alice finds out that one of her friends, Caroline, gathers the information about digital libraries and her expertise level on that topic is very high. Though her direct friend Bob is interested in Artificial Intelligence, she finally decides to link resources provided by Eric, who has a highly ranked "Semantic Web" collection. From now on, Alice takes the advantage of the information gathered by Caroline and Eric in their collections.

Secured Semantic Social Collaborative Filtering Alice is still looking for more information required for her thesis. She decides to register in an open, hererogenous digital library. Some people protect their collections with access control restrictions (see Fig.3). The restrictions applied on the collection are based on maximal distance and minimal trust level between two people in the social network graph. Apart from defining friendship relations, users express the quality (trust level) of every outgoing social connection.

Since not all information should be accessible by everyone, some of it need to be protected from people from the outside of the given community. This is why access control lists (ACL) have been introduced (see Fig.4(a)). In the semantic social collaborative filtering environment based on ACL each collection has its own ACL, that defines the maximal distace and minimal friendship quantisation level from the specific person[1] to the person willing to access that collection. Only when this is satisfied the user can access and include this collection in his/her collections.

Alice wants to make use of the knowledge provided by Damian. But the algorithm for retrieving a list of collections in the secured environment (see Fig.4(b)) omitted some of collections. With ACL applied Alice is out of the range defined in Damian's ACL constrains. The collection managed by Damian is not presented to her.

[1] please note that it does not have to be an owner of the collection, thought the owner is the one that manages ACL

procedure ListCollections$_{SM}$(p,t) : collections[]
for $p' \in P$ **with** `PeerDistance`(p,p') $<$ *knowsRange*
$C' \leftarrow C' \bigcup$ `PeerCollection`(p')
end for
sort C' according to `FinalRanking`$_{SM}$
end procedure

(a) Algorithm retrieving list of collections

P is a set of peers
C is a set of collections
$FoafKnows$ is a set of directed connections between peers
$G_{peers}(P, FoafKnows)$ is a digraph of friendship relations
T is a lattice of categorisation topics
We assume that each collection $c \in C$ has exactly one owner $p \in P$.

`PeerCollection`: $P \rightarrow 2^C$ – returns all collections owned by the peer
`OwnedBy`: $C \rightarrow P$ – returns the owner of the collection
`Expertise`: $(P, C) \rightarrow [0, 1]$ – computes the quality of the collection based on the peer's expertise on related topic
`Categorisation`: $C \rightarrow T$ – returns the list of topic describing collection
`PeerDistance`: $(P, P) \rightarrow \mathbf{N}$ – computes distance between two peers in the social network graph using Dijkstra algorithm
`Similarity`: $(T, T) \rightarrow [0, 1]$ – computes similarity level between two topics
`FinalRanking`$_{SM}$: (`PeerDistance`,`Similarity`,`Expertise`) $\rightarrow [0, 1]$ – computes ranking value for a collection based on distance to the owner, similarity level and quality measure

knowsRange – defines a maximal distance between two people when traversing the graph of friendship relations.

(b) Definition of model

Fig. 2. The simple model of social collaborative filtering

2.3 The Benefits

The main bottleneck of existing passive collaborative filtering systems is the process of gathering users' preferences[4]. A reliable system requires a very large number of people to express their opinion about a large number of topics. This requires from users to either fill out a survey or perform some activities (like e.g. buying a product, reading a book) over a certain time.

Active collaborative filtering solutions depends on maintaining the social network by users themselves. Outdated information on list of friends can mislead the person in his quest for an answer.

Backward Referral Chaining Maintaining a list of friends, posting a question and gathering the answers may be time consuming. That is why the social collaborative filtering (a new approach to active collaborative filtering) tends to

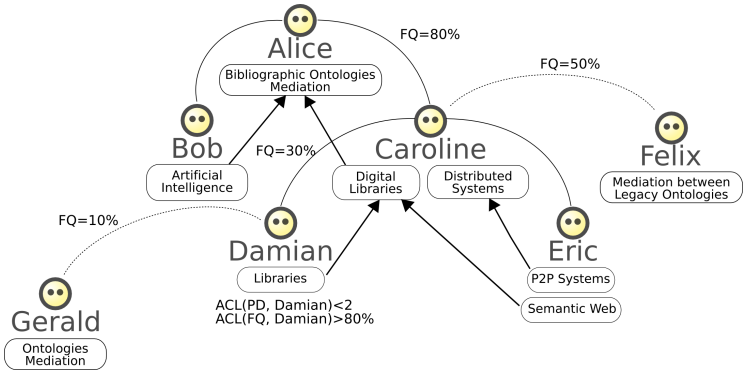


Fig. 3. The scenario in the secured semantic social collaborative filtering model

ease some hardships by introducing the concept of *backward referral chaining*, reusing existing classification schemata and extrapolating user's profile information with interests of his friends.

Usually, a user is not aware of the whole social network. To gather the knowledge outside of his direct friendship neighbourhood the user has to rely on references provided by his friends. Because the expert in the specific domain can be quite distant from the user, in terms of relationship links, the access to the answer provided dependents on the path to an expert. As it has been introduced in 2.2, an expert can restrict the access to some parts of information by applying access control lists.

The referral chaining[11] has two strong dependencies: *accuracy* of finding the right path to an expert, and *responsiveness* factor of the found expert. The backward referral chaining introduced in the social collaborative filtering inverses the process of finding an expert. The answers provided by different people (including experts) are being assembled into hierarchical knowledge base. Users link into their collections, information provided by some other people. In many cases, the expertise of the latter, on given topic is higher.

Connection to the established classification schemata. In social collaborative filtering each person can create own categories according to the local understanding of the world. The definition of the category might be hard to understand to other peers because of the use of ambiguous descriptions or an native language.

We propose to apply additional semantically reach description based on existing thesauri or classification ontologies, like WordNet[12] or Dewey Decimal Classification[13, 9]. This description can help to understand the meaning of the category both to people and machines. The latter can then utilise this knowledge in e.g. recommending related categories created by other peers.

ACL_{PD} is an access control constrains, defining maximal distance D (in number of 'hops') from user P
ACL_{FQ} is an access control constrains, defining minimal $FriendshipQuantization$ value (calculated across the graph) from user P

`DistanceACL`: $(C) \rightarrow 2^{ACL_{PD}}$ – defines a list of allowed maximal distances to the user
`QuantizationACL`: $(C) \rightarrow 2^{ACL_{FQ}}$ – defines a list of allowed minimal $FriendshipQuantization$ values
`Peer`: $(ACL) \rightarrow P$ – returns a peer from which the computation of ACL dinstance/level is do be performed
`Distance`: $(ACL_{PD}) \rightarrow \mathbf{N}$ – returns the maximal distance defined in ACL
`Quantisation`: $(ACL_{FQ}) \rightarrow [0,1]$ – returns the minimal $FriendshipQuantization$ level

(a) Definition of model

procedure ListCollections$_{ACL}$(p,t) : collections[]
$c_p \leftarrow$ `PeerCollection`(p)
for $p' \in P$ **with** `PeerDistance`$(p,p') < knowsRange$

for $c' \in$ `PeerCollection`(p')
with $\forall acl_{PD} \in$ `DistanceACL`(c') `PeerDistance`(`Peer`$(acl_{PD}), p) <$ `Distance`(acl_{PD})
with $\forall acl_{FQ} \in$ `QuantizationACL`(c') `FriendshipQuantization`(`Peer`$(acl_{PD}), p) >$ `Quantisation`(acl_{PD})
with `CollectionDistance`$(c_p,c') < inclusionRange$
$C' \leftarrow C' \bigcup \{c'\}$
end for
for $c \in C$ **with** `FriendshipQuantization`$(p,$`OwnedBy`$(c)) > quantisationLevel$
if $\forall acl_{PD} \in$ `DistanceACL`(c') `PeerDistance`(`Peer`$(acl_{PD}), p) <$ `Distance`(acl_{PD})
and $\forall acl_{FQ} \in$ `QuantizationACL`(c') **and** $\exists c' \in C'$ `CollectionDistance`$(c_p,c') +$ `CollectionDistance`$(c',c) < inclusionRange$
then $C'' \leftarrow C'' \bigcup \{c\}$
end for
sort $(C' \bigcup C'')$ according to `FinalRanking`$_{CI}$
end procedure

(b) Algorithm retrieving list of collections

Fig. 4. The secure model of social collaborative filtering

Extrapolated user's profile. When information about user's activities (personal bookshelf, resources' annotations) is gathered for a longer time it can be re-used during the search process. The query expansion process[9] takes into account semantically rich descriptions of users' preferences reflecting their activities. The result set becomes more user oriented than with a generic search.

New users registered to the system very often suffer from lack of rich profile information. This may have a strong influence on the quality of search results. To overcome this problem then social collaborative filtering paradigm introduces the concept of *an extrapolated user's profile*. The profile of the new user can

be represented with some probability depending on trust level (see 2.4) as a combination of profiles of his/her friends.

2.4 Security and Privacy Issues

Collaborative filtering implementations suffer in most cases from very weak security features or frequent privacy abuse. The information about the user in passive collaborative filtering systems is very often gathered without his knowledge. In the active collaborative filtering the user very often has no chance to protect himself from gathering information about him.

To implement the security and privacy features the concept of digraph of interpersonal connections have been utilised. Each user defines a list of his friends and states the level of trust to each of them. The user can then define the maximal distance and minimal trust level required from the person which wants to view information gathered in specific category.

As all the information about the user is provided by himself and he/she manages the access control lists for each piece of information, the privacy of the user is preserved.

3 Evaluation of Semantic Social Collaborative Filtering

Semantic social collaborative filtering utilises existing social networks instead of creating artificial connections between people. That is why on the contrary to other collaboration filtering solution, there is no need to evaluate an algorithm for creating a social network, as the social network is given explicitly.

On the other hand, since the semantic social collaborative filtering is based on friendship connections, the actual similarities of interests between connected users might differ. That is why, the evaluation of this collaborative filtering approach should prove that the dissemination of knowledge is possible within graph of semantically annotated friendship connections.

Simulation model In this section we present the implementation of the simple semantic social collaborative filtering model. We prove that average level of expertise in the subgraph of social network is almost maximal within 6 degrees of separation.

The definition of a simulation model has been based on similar ideas defined in Refferal Web project[11]. The main difference between social semantic collaborative filtering and the Refferal Web is that in the Refferal Web project, the process of finding an expert on certain topic is performed manually by the user. In semantic social collaborative filtering, semantical annotation on the knowledge provided in the social network is used to automate the process of finding the high quality of information. The simulation model itself might be similar to the one presented in [11], so we just need prove that it is possible to find an expert on the given maximal degree of separation.

Underlying assumptions. In the model of social network for the semantic social collaborative filtering, each user manages collections with information on selected topic. The different users represent different expertise on the given topic. We assume that:

- The quality of the information provided by a user on a certain collection is proportional to the expertise level of the user on the topic of collection.
- It is possible to find a user with a high expertise on given topic within the network of social connections.

According to simple social collaborative filtering model (see Fig. 2(b)) the simulation environment is modeled by a set of users and a set of collections managed by those users. There is exactly one user that owns each collection. On the base of the user's expertise on related topic the quality of the collection is defined. Each user has a predefined set of other users he knows (this relation should not be considered as implicitly symmetric).

Although according to the Small World Phenomena[14, 15] the distribution of the degree of the friendship connections is power-law based (Zipf's distribution, see Eq. 1) we have decided to perform second set of experiments where the degree of friendship connections is a bell-curve shaped (normal random variable see Eq. 2).

$$O(i) = \frac{n}{i^{\theta} H_{\theta}(V)}, H_{\theta}(V) = \sum_{i=1}^{V} \frac{1}{i^{\theta}} \tag{1}$$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

The distribution of expertise on a certain topic within the social network can be based on the Lotka's Law[16], stating that *the number of authors making n contributions is about* $\frac{1}{n^a}$ *of those making one contribution, where a is often nearly 2*. Since the expertise on a certain topic is proportional to the number of high quality of publications, the probability of the level of expertise (the level of expertise over the number of users that have one) is Zipfian shaped as well. Each collection has a quality value assigned to it that represents the expertise the owner has on the related topic.

In order to make sure that there would be at least one absolute ($\texttt{Expertise}(T) = 1$) expert in each topic T, we have normalised the associated expertise values dividing each but the value of the highest expertise in each topic.

The list of topics used to describe content of collection has been based on Dewey Decimal Classification[13]. This simplifies the computability of the model in the sense of comparison similarity between topics. Each category has a three-digit number (100 - 999) associated. Categories are structured as a three level tree. Although in the real world implementation categories are described additionally with WordNet words vectors, DDC seems to be enough for the modeling purpose.

```
procedure AverageMaximalExpertise(R) : E⁻max(R)
 for p′ ∈ P with
  select t ∈ Tp find c that
   t =Categorisation(c)
   PeerDistance(p, Owner(c)) < R
   e =Expertize(Owner(c), c) is maximal
  AverageMaximalExpertise + = e/NP
end procedure
```

Fig. 5. Algorithm calculating average maximal expertise in the semantic social collaborative filtering model in the given range

Definition of the experiment. During the experiment each user ($p \in P$, $\texttt{sizeOf}(P) = N_P$) tries to find in the social network within a given range R, the collection that provides the information on the topic $t \in T_p$. The topic is randomly selected from the list of topics associated to collections owned by the user. The average value of the highest expertise $\bar{E}_{max}(R)$ level found within given range is computed (see Fig.5).

We have performed four experiments. Each time the social network model consited of $N_P = \mathbf{1000}$ users.

Each user in our social collaborative filtering environment had only one collection associated. This simplification is correct since during the experiment we are looking only for collections with exactly the same topic as selected. So collections associated with each topic creates a subgraph that is independent of the actual number of collections owned by each user.

The expertise level for each collection has been randomly selected according to power law distribution. In the first two experiments the degree of friendship connections has been randomly selected according to normal distribution (μ =**25**, σ =**12.5**). In the last two experiments the power law distribution (θ =**1.9**) has been applied. During each experiment average maximal expertise values $\bar{E}_{max}(R)$ has been calculated for maximal degree of separation $R \in [1, 8]$.

Results of simulation. Table 1 presents results of all four experiments.

Table 1. Results of the experiment - average maximal expertise $\bar{E}_{max}(R)$

R	**σ = 12,5F (Bell)**		**θ = 1,9F (Zipf)**	
1	0,07072	0,06427	0,01793	0,01595
2	0,69098	0,69192	0,10557	0,09042
3	0,96399	0,96183	0,33044	0,29836
4	0,96796	0,96782	0,62892	0,61653
5	0,96796	0,96782	0,82896	0,82980
6	0,96796	0,96782	0,90953	0,91751

It is interesting that even for the power law based distribution user is able to find information with almost the highest possible quality within **6** degrees of separation (see Fig.6).

Conclusions on results of simulation Following experiments by Kauth[11] we have constructed similar social collaborative filtering model. The results revealed that each user is able to find (on average) the best quality of information provided by other users within the subgraph of social network bounded by **6** degrees of separation. These experimental results proved that the constructed social network model corresponds to the small world phenomena[14]. Hence, the assumptions underlying the social collaborative filtering has been fullfiled. It is possible to find an expert (with an average expertize level above 90%) within the small social network neighbourhood.

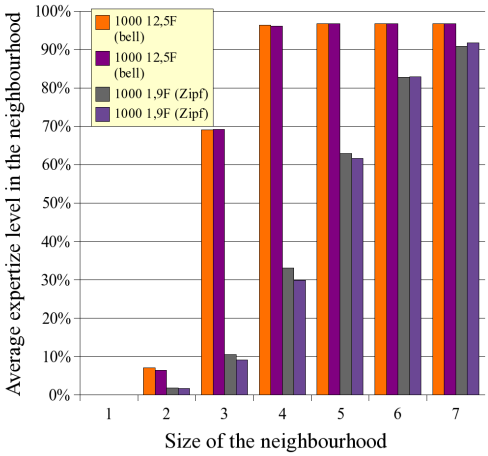


Fig. 6. Average expertise level in the neighbourhood of the given size

4 FOAFRealm - the Reference Implementation of Semantic Social Collaborative Filtering

The FOAFRealm is a library for distributed users management based on the FOAF vocabulary. It enables users to control their profile information, as the information can be accessed in the open FOAF format. Users can sign-in automatically across the P2P network (called D-FOAF[2]) of FOAFRealm enabled systems[17].

FOAFRealm provides a basic implementation of the semantic social collaborative filtering concept. The knowledge (annotations and private collections) can be shared among registered users. Security constraints can be applied to each piece of information separately.

The current implementation of FOAFRealm consists of four layers:

[2] D-FOAF project: http://d-foaf.foafrealm.org/

– The distributed communication layer providing access to highly scalable HyperCuP[3] P2P infrastructure to communicate and share the information with other FOAFRealm implementations..
– FOAF and collaborative filtering ontology management. It wraps the actual RDF storage being used from the upper layers providing simple access to the semantic information. The Dijkstra algorithm for calculating distance and friendship quantisation is implemented in that layer.
– Implementation of the `org.apache.catalina.{Realm,Valve}` interfaces to easily plug-in the FOAFRealm in to Tomcat-based web applications. It provides authentication features including autologin based on Cookies.
– A set of Java classes, Tagfiles and JSP files plus list of guidelines that can be used while developing user interface in own web applications.

The library has been successfully deployed as a user management system in JeromeDL - e-Library with Semantics[4]. It is used to handle private bookshelves of readers, and provides additional semantical annotations to the resources. The concept of extrapolated user profile has been adapted in the semantically enhanced search engine in JeromeDL. So that even new users to the system can benefit from the full-fledged semantic search process.

The FOAFRealm system has also become a part of MarcOnt Initiative[5] collaboration portal for ontologies management based on negotiations. The portal will utilise social networks based features of FOAFRealm to:

– *isolate outside world from the ontology management community.* The registered users will be allowed to take part of the ontology management process when they will be defined as a friend of at least on of the community members.
– *differentiate evaluations of ontology changes suggestions provided by different members of the community.* We will explore if evaluations provided by close friends of the person that posted the suggestion should be ranked lower than evaluations provided by people with higher degree of separation from the suggestion owner.

5 Related work

Collaborative filtering. The most popular types of the collaborative filtering systems are Active Collaborative Filtering and Passive Collaborative Filtering. The distinction between those is based on the activeness of the user that receives information. With passive collaborative filtering, the information about the user, such as: mailing-lists posts, links on home pages, citations in publications and co-authors of articles, is utilised. Since the user does not actively take part in maintaining his network of friends, he has no direct impact on information he/she receives.

[3] Lightweight HyperCuP Implementation project: http://www.hypercup.org/
[4] JeromeDL - e-Library with Semantics: http://www.jeromedl.org/
[5] MarcOnt Initiative: http://www.marcont.org/

Active collaborative filtering implements two models of information retrieval: *user pull model* - where a user generates a query to the network of other users, and *user push model* - where the answers on previously stated questions or information filters, are feed to the user.

Though by shifting from central (a search engine) to a distributed method of recommendation the problem tends to be more manageable, particular collaboration filtering implementations suffer various difficulties: **(1)** "heterophilous diffusion" (exchange information across different socio-economic groups) is neglected in favour to "homophilous diffusion" (exchange of information within socio-economic groups); **(2)** security and privacy issues are weakly supported; **(3)** meaning (semantics) of shared concepts are lost; **(4)** when the network of friends is created automatically by harvesting various databases with advance algorithms: the "critical mass" of registered users is required to provide satisfiable level of correlation to user's interests; it is impossible to create a digraph of social connection from most of commonly used sources; privacy of individuals is violated; monopolies are supported[18] because a service provider has to gather a lot of information to become accurate ("critical mass"); **(5)** when the user actively uses fora or mailing-lists: (i) there is no guarantee that there will be an answer to the posted question, or that the answer will be through; (ii) there might be no expert on the specific field of discourse in the "direct friendship neighbourhood" of the user; **(6)** some systems requires from users to answer long questionnaires [4] in order to find similarities in users' interests.

Hybrid filtering[19], the combination of content filtering and social filtering, is used to maximise precision with a recall still above specified limit.

Active collaborative filtering solutions concentrate on utilising the existing social connections provided explicitly. One of the approaches [10] is build on the the common practise where people tell their friends or colleagues of interesting documents. Users collect bookmarks on the interesting World Wide Web pages that they have found. [20] describes a social collaborative filtering system where users have direct impact on filtering process. The changes in the users interests are exploited to provide thorough relevance feedback to the system..

To format and distribute collections of bookmarks several simple system have been developed. With Simon system [21] users can create "subject spaces" which are lists of hypertext links to the WWW pages with annotations on them. One of other possible solutions is to find a personal referral that can answer the given query. The network of relationships can also help in exploring the hidden web, the part of the Internet that is not indexed by search engines [11], as some of the information is deliberately not accessible outside the intranets [10, 22].

Online social communities are the underlying key concept of the semantic social collaborative filtering presented in this article. In the last few years this field has been widely explored by several implementations.

Some of them, like Orkut[6] provides forum-like channels of dissemination of knowledge, where community members can ask questions to their friends

[6] Orkut online community portal: http://www.orkut.com/

or other members of specific thematic group. In the Semantic Web field the FOAF (a vocabulary for RDF [23]) format has been introduced to describe the interpersonal connections.

User Profile Management The existing implementations of user profile management lack: (i) fine granularity of security constraints; (ii) scalability; (iii) openness/privacy. Both of which play important roles in semantic social collaborative filtering.

One of the features that is becoming more and more important in social P2P environment is single-sign-on[24]. Each time a user uses a new web system, he would rather not provide all the same information about himself over and over again. Solutions like Microsoft Passport[7] or Sxip[8] provide such features.

6 Future Work and Conclusions

The semantic social collaborative filtering presented in the article opens new possibilities of exchanging and managing knowledge. Users can share their bookmarks (collections and their content) with their friends. Everyone can organise the knowledge by gathering collections that other people are maintaining. Since collections can be linked it is possible to find more relevant information in categories provided by some distant people. Annotations are also a key part of the semantic social collaborative filtering. Together with private collections (private bookshelves) they are utilised in the semantically enhanced information retrieval in systems like digital libraries.

FOAFRealm is a reference implementation of the semantic social collaborative filtering. It refers to social networks and open standards like FOAF. FOAFRealm provides support for J2EE based web applications for quick extending their features with user management and social collaborative filtering. Since the social network is represented as a digraph, FOAFRealm utilises informations about distance between two people and the trust level, to provide the security and privacy features.

Current implementation of FOAFRealm, D-FOAF, provides a distributed user profile management system and hence, the social semantic collaborative filtering across different systems. The future step, DigiMe, will deliver this features to mobile devices and will explore the ad-hoc social networks paradigm.

References

1. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35** (1992) 61–70

[7] Microsoft Passport: http://www.passport.net/
[8] SXIP - Passport/Liberty done right: http://www.sxip.com/

2. Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, ACM (1994) 175–186
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. (1995) 43–52
4. Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating "Word of Mouth". In: Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. Volume 1. (1995) 210–217
5. Trevor, J., Hilbert, D.M., Billsus, D., Vaughan, J., Tran, Q.T.: Contextual Contact Retrieval (2002)
6. Fukui, H.O.: SocialPathFinder: Computer Supported Exploration of Social Networks on WWW (2003)
7. Newman, M., Watts, D., Strogatz, S.: Random graph models of social networks. In: Proc. Natl. Acad. Sci., to appear. (2002)
8. Hoadley, C., Pea, R.: Finding the ties that bind: Tools in support of a knowledge-building community (2002)
9. Kruk, S.R., Decker, S., Zieborak, L.: JeromeDL - Reconnecting Digital Libraries and the Semantic Web. In: DEXA'2005. (2005)
10. Maltz, D., Ehrlich, K.: Pointing the way: active collaborative filtering. In: Proceedings of the Conference on Computer-Human Interaction. (1995) 202–209
11. Kautz, H.A., Selman, B., Shah, M.A.: The Hidden Web. AI Magazine **18** (1997) 27–36
12. Fellbaum, C.: WordNet An Electronic Lexical Database (1998)
13. Dewey, M.: A Classification and Subject Index for Cataloguing and Arranging the Books and Pamphlets of a Library - Dewey Decimal Classification. guternberg.net, http://www.gutenberg.net/catalog/world/ readfile?fk_files=59063 (2004)
14. Milgram, S.: The small world problem. Psychology Today **67** (1967)
15. Barabasi, A.L.: Linked: The new science od Networks. Cambridge Perseus Press (2002)
16. Lotka, A.J.: The Frequency Distribution of Scientific Productivity. Journal of the Washington Academy of Sciences **16** (1926) 317–323
17. Grzonkowski, S., Gzella, A., Krawczyk, H., Kruk, S.R., Moyano, F.J.M.R., Woroniecki, T.: D-FOAF - Security Aspects in Distributed User Managment System. (In: TEHOSS'2005)
18. Techniques, R.P.: Privacy-Preserving Collaborative Filtering using (2003)
19. Basu, C., Hirsh, H., Cohen, W.W.: Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In: AAAI/IAAI. (1998) 714–720
20. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. In: WWW '04: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, ACM Press (2004) 675–684
21. Simons, J.: Using a Semantic User Model to Filter the "World Wide Web" Proactively. (1995) 455–456
22. Kautz, H., Selman, B., Shah, M.: Referral Web: Combining Social Networks and Collaborative Filtering. Communications of the ACM **40** (1997) 63–65
23. http://www.w3.org/TR/rdf: Resource Description Framework (RDF). W3C Recommendation (1999)
24. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A Community Authorization Service for Group Collaboration (2002)

Task Specific Semantic Views: Extracting and Integrating Contextual Metadata from the Web

Stefania Ghita, Nicola Henze, Wolfgang Nejdl

L3S Research Center / University of Hanover
Deutscher Pavillon, Expo Plaza 1
30539 Hanover, Germany
{ghita,henze,nejdl}@l3s.de

Abstract. Tasks and working scenarios on the desktop involve specific context information which is useful for finding relevant documents related to that context. Automating the process of retrieving and generating this context information is important to avoid time-consuming manual annotation not feasible in everyday work. This paper focuses on automatically extracting and integrating contextual information from web pages used in such working scenarios. The key observation is that in such scenarios we often use a set of web sites to get relevant information, implicitly syndicating their data into a coherent scenario specific information space. We show how these data can be extracted automatically from the web pages stored in local browser caches, based on appropriate query wrappers over these pages. These data are then combined into a task specific semantic view, building upon schema integration rules based on a global as view approach and view materialization, and transformed into RDF metadata for enhancing contextualized search on the desktop. We describe both the conceptual framework as well as our current prototype and conclude with a discussion of further research issues.

1 Introduction

People structure their (work) lifes according to their main activities and, emerging from these daily activities, browsing history is an important mirror of their information seeking behavior. Typically, when people search for information on the web, they do not rely on only one source of information, but many. For example, if a user searches for a publication and its relevant context, he does not only search on one web site, but will combine the information from several sites into a coherent whole. For example, on CiteSeer he will look for the papers that are cited by a specific paper as well as the ones citing it, and on DBLP he will look for the conference that the paper was published at and search for more papers in the same track. So in general, what people try to do is to collect useful information from many sites and manually syndicate this information on their desktop, hoping to have a better view over all relevant information available for specific tasks.

This information is very useful in the desktop search context, as we have discussed in [6], and is available in browser caches. However, from these pages stored as HTML documents, it is very difficult to extract the relevant information automatically. What

we really need is to have this information represented in a structured form, automatically transformed into the relevant task specific RDF context metadata, whose structure is specified using task specific ontologies. The main contribution of this paper is to show how this can be done, based on automatic extraction of relevant context information from web sites and automatic syndication of this context information into context metadata specified by task specific ontologies representing a global view over available context information. We will present an integrated system for context extraction and syndication based on the user's browsing behavior, that is able to gather information from web sites visited by the user during his activities. The system not only stores web pages in their original form, but extracts and syndicates all relevant information, thus reconstructing the relevant information space underlying these pages.

In the next section we present two motivating scenarios and discuss how we extract and syndicate information in such task specific scenarios. We will discuss in more detail in Section 3 the schemas describing web data sources and task specific ontologies, as well as the transformation steps required to extract relevant information and to syndicate it into (materialized) task specific global views. These transformation steps are further detailed in 4, where we describe how we extract web information using the Lixto toolkit and how we materialize our task specific views using mapping rules written in the TRIPLE language. We also describe how these transformation steps are automatically triggered in our Beagle^{++} desktop search infrastructure. We conclude with a discussion of related and future work.

2 Motivating Scenarios

Let us start with two working scenarios where we want to retrieve already viewed information, the first one suitable for research activities, the second suitable for movie fans.

2.1 Research Scenario

Alice is a researcher that has as her main interests peer-to-peer networks and RDF technology. Some time ago she searched for some papers about this subject on the web. These actions have been memorized and will influence her personal profile.

In order to find the necessary information, Alice uses two main information sources: CiteSeer, *citeseer.ist.psu.edu*, for the citing and cited papers, and DBLP, *http://www.informatik.uni-trier.de/~ley/db/*, for the conference information (tracks, editions). Alice discovered a paper about Edutella, **"EDUTELLA: A P2P Networking Infrastructure Based on RDF"** on the CiteSeer web site, as well as some other papers that cited this one, including ones that were written by the same author, Wolfgang Nejdl. As Alice was interested in the conference that this paper was presented at, she followed the link towards the DBLP web site. On the corresponding WWW 2002 conference page on DBLP, she looked at another paper in the same track (Query Language for Semantic Web) as the Edutella paper, *"RQL: a declarative query language for RDF"*. Another paper published by the same author was available from the WWW 2003 conference, ***"Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-to-Peer Networks"***.

The system has stored all this browsing behavior and represented it as RDF metadata, both from the CiteSeer web page and the DBLP one. The data harvested this way contains all data from the appropriate web pages dedicated to a certain paper or conference. This merged data are very useful, as it provides all relevant information about each resource, as seen in Figure 1.

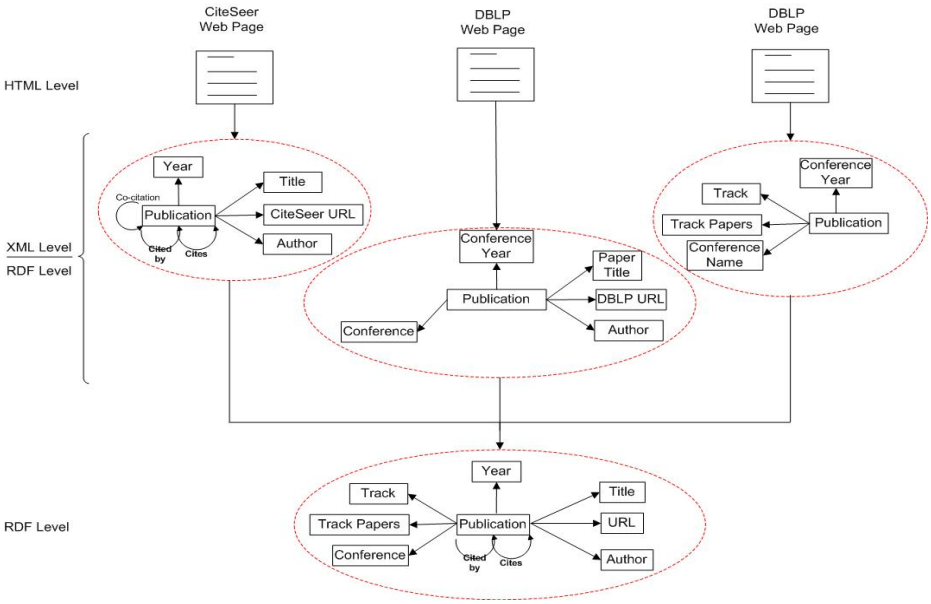


Fig. 1. Syndicated Data View for Research Scenario

When Alice searches again for Edutella resources, she does not only find the paper she retrieved, but additionally get the stored context of this paper. In particular, this includes some publications that cited this paper, and especially the ones with the same author: ***"Super-Peer-Based Routing Strategies for RDF-Based Peer-to-Peer Networks"***, ***"Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-to-Peer Networks"***, ***"Role Oriented Models for Hypermedia Construction"***. The system is also able to make the connections between different resources based on the research scenario context metadata. The system knows that Alice viewed the paper both on CiteSeer as well as on the DBLP site, and displays additional data that was extracted from that web site, including the conference that the paper has been presented at, some other papers that were presented in the same track, and other editions of the same conference (WWW).

2.2 Movies Scenario

In our second scenario we investigate how the context is useful for movie fans. Let us take a look at Bob who is a big fan of Harrison Ford and likes science-fiction movies. He

is usually browsing the IMDB (*www.imdb.com*) web site for information and his personal history includes that that he has viewed Harrison Ford's page so many times. For purchasing products from the internet, Bob uses Amazon (*www.amazon.com*), which also has links to the IMDB site.

Some time ago, Bob ordered the "**Star Wars**" CD. Now he wants to buy the DVD, too, and he will be provided with the relevant context. When searching for this DVD, the system realizes that Bob has in the past browsed information about this movie and displays the joined information from the two intensely visited web sites, Amazon and IMDB. For example, the director and the actors for this movie will be displayed, together with other movies that these actors participated. Since Bob is a big fan of Harrison Ford, whenever he views the page on Amazon about the "Star Wars" movie, he will automatically see information retrieved from the web pages of the movies that Harrison Ford participated, as viewed on IMDB. Among the displayed metadata, Bob can also see the rating from IMDB (7.7/10) or the price that he paid for the CD at the time he had bought it. He will also receive some additional movie recommendations from IMDB ("Shaft") and from Amazon ("Psycho").

Besides these suggestions, Bob will also receive additional valuable information, i.e. comments from users of the IMDB web site. Not any recommendations, though, since the system will be able to choose among the multitude of comments about this movie the one made by his favorite commenter, "*Grann-Bach*". This comment is also very highly rated by the users of the IMDB system (22/30), even though on the IMDB page some other user's comment would appear. This will be done because Bob always reads his comments and this fact is memorized in his personal history. So whenever Bob wants to buy something, additional information about the product will be made available to him, based on information extracted from his past activities (see Figure 2).

3 Relevant Data and Transformation Steps

As seen in Figure 2, we can partition the process behind our motivating scenarios into distinct steps, further detailed in the rest of this paper. The first step takes care of the extraction of information from various web sites, each web site having a specific schema for their content, as discussed in Section 3.1. The data retrieved in an XML format will contain the relevant web page information for each context and will be transformed into RDF data using XSLT (Section 3.2). After this transformation, the data are syndicated and materialized (based on appropriate mapping rules) into task specific semantic views, and can be used to answer queries based over these views.

3.1 Schemas for Web Page Content

Data driven HTML pages contain two kinds of information: about the structure of the page (repetition of items) - data items are listed as rows of a table, or are structured in distinct sections - and about what is presented within this structure - the actual information. The first type (partially) reflects the structure of the database that were used to generate the web page. All pages generated from databases and a lot of other ones repeat the same structural items so that we can recognize different information items

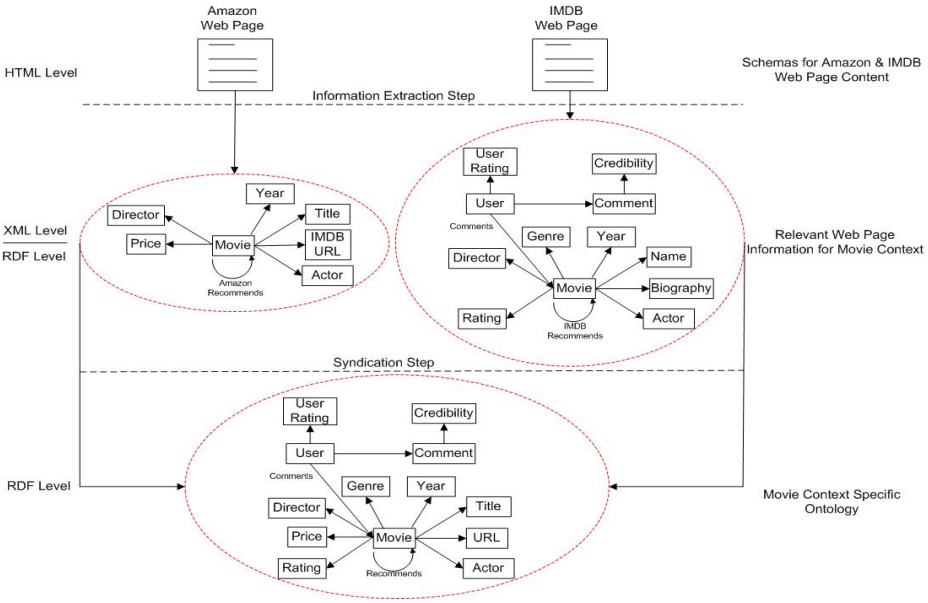


Fig. 2. Syndicated Data View for Movies Scenario

rather easily. For example, a Hotel Information Server has web pages structured based on the databases entries: the description of an individual Hotel with details like general information, pictures, contact address, etc. In the case of our research scenario, the information about scientific publications is often presented in the same style (see Figure 3), and includes title of the paper, year of publication, authors, and cited papers.

As a first step, we need local schemas for the web pages interesting for a specific scenario. Appropriate collections of web pages share some structure for presenting the content, e.g. all pages from CiteSeer that present information about a publication belong to a class, together with the web pages that are associated to this page by dedicated links (in Figure 3 these are the links under the rubric "cited by" plus the according web pages). In a (manual) preparation step, we analyze each of these collections for expressive metadata, and design a small, local ontology which describes the objects of discourse of each of these collections. We can then harvest information about entities and their different attributes. For example, for papers we have publications and conferences and the attributes for these entities. Similarly, the IMDB and Amazon web sites reflect the underlying database entries: movies, actors, directors. Such a local schema for the CiteSeer source is depicted in Figure 4.

In all these cases, the information on the web page is semi-structured, and this will allow us to construct machine-readable metadata from these web pages in a semi-automated manner (see Section 4.1). This is done based on the reconstructed schema and an appropriate query on the HTML page which extracts information according to that schema.

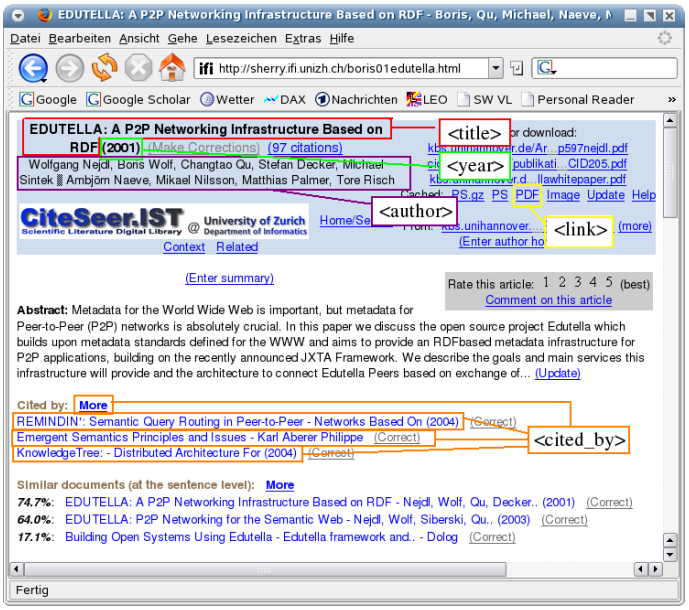


Fig. 3. CiteSeer Web Page

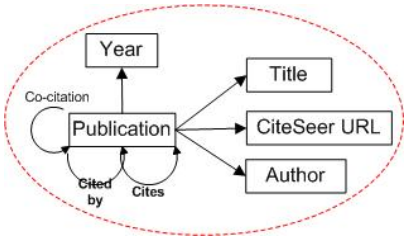


Fig. 4. CiteSeer Local Schema

3.2 Task Specific Semantic Views

Depending on the tasks and context the user is working in, his context includes all relevant information from the local schemas which we described in the last section. This context can be represented by a task specific semantic view integrating the relevant data from the local sources we discussed in the previous section. This semantic view specifies all metadata needed of this context, and is described using ontologies. Let us take a look on two ontologies appropriate for our two example scenarios.

Research Ontology. Figure 5 depicts an overview image of the ontology that defines appropriate context metadata for the research scenario. The ”Publication” class represents a specific type of file, with additional information associated to it. The most important attributes are “Author”, “Title”, as well as relationships regarding citing and

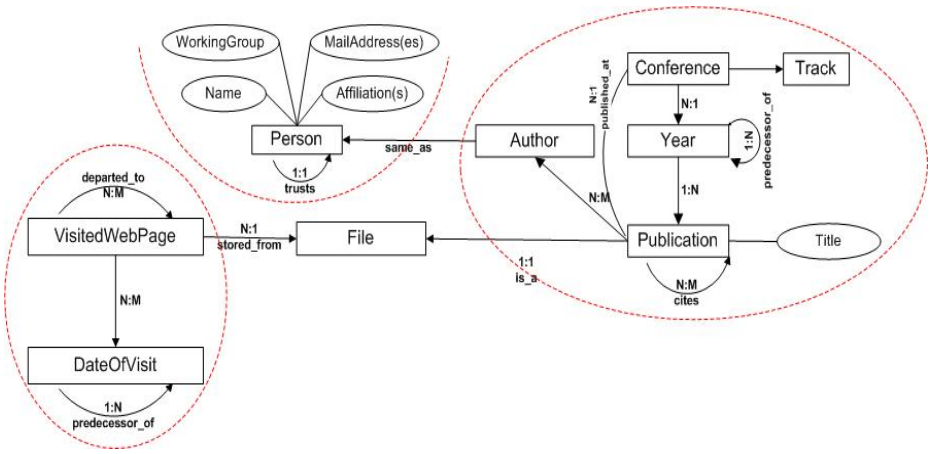


Fig. 5. Research Ontology

cited papers. These attributes and relations can easily be retrieved from a CiteSeer web page, "Conference" and "Year" harvesting is done more reliable from the DBLP site. As each publication is stored as a file, it is also connected to the file context, and thus to the file specific information like path, number of accesses, etc. Additionally, it is possibly connected to visited web pages / URLs a publication has been downloaded from. Authors are persons, which are modeled based on the FOAF ontology, as members of interest groups (foaf:Group).

If we take a look at Figure 1 we see that some attributes are omitted from the more web page-specific ontologies, such as "Co-citations" retrieved from the CiteSeer web page, and that some attributes have different names, even though they represent the same information, such as "Title" and "Paper Title". So this ontology represents a specific view on the local data sources, appropriate for the task context.

Movies Ontology. The central part of the movies ontology is the "Movie", as seen in Figure 6. It is associated to the participating actors, the director, the year it was released and its genre (comedy, action, thriller, science fiction, etc.) The IMDB site also provides a "Rating" computed with the help of the ranks provided by different registered users. An interesting feature of this database is the fact that the system also makes recommendations for other movies. Users comment on movies, these comments have a credibility value based on their usefulness to other users. As in the research scenario, this ontology again represents a syndicated view of the Amazon and the IMDB ontologies, as seen in Figure 2.

3.3 Transforming Web Page Content Information into Task Specific Metadata

Combining data from different sources and providing the user with a unified view of these data is known as the "data integration" problem [19, 16, 4]. The set of sources (in our context the set of visited web pages) contain the relevant data, while a global schema

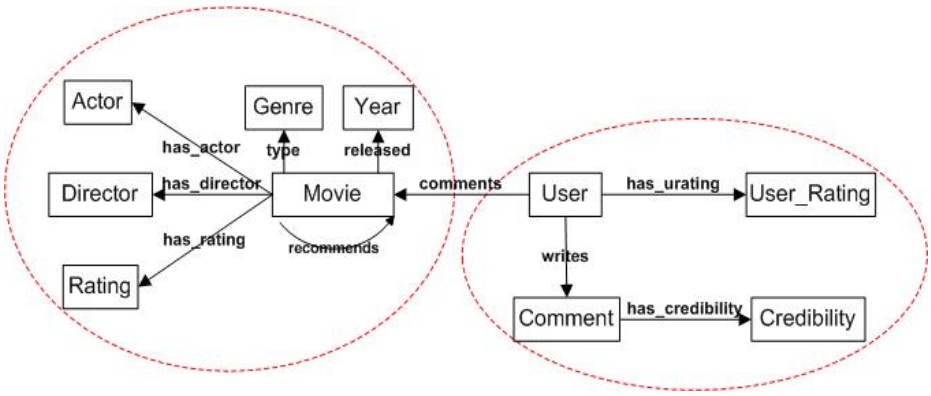


Fig. 6. Movies Ontology

(in our context the task specific ontologies) provide a unified view of the underlying sources. For modeling the relation between the sources and the global schema we will use the global-as-view approach [8, 10], which describes the mappings between local sources and the global schema as a set of assertions

$$g \rightsquigarrow q_S$$

where g represents an element of the global schema and q_S a query over the sources. Such a mapping explicitly specifies how to query the local sources for each element contained in a query over the global schema, or alternatively, how to materialize the global schema based on the instances from the local data sources.

Using these mappings, we can materialize instances of the task specific ontologies when the user browses new web pages, or reformulate queries over the task specific ontologies during search time. Obviously, the second alternative is not really useful in our context as users have come to expect nearly instantaneous access to search results from web search engines.

As we assume exact views and do not allow integrity constraints in the global schema, our data integration algorithm can exploit the "single database property" [4] which means that all instances of relations of the global schema can be computed by the corresponding views over the sources, using the mapping rules as transformation rules. So view materialization is the more feasible approach for our application, and we just have to remember how the derived global schema instances depend on the source data in order to recompute the views when source data change. This is similar to view materialization in data warehouses [7].

The global database is thus constructed by merging the important information from the relevant sources of information, i.e. the information extracted from the web pages browsed are merged into the global database containing our activity driven metadata as specified by the task specific ontologies. We can easily see that the data is not only a projection or a subset of the data provided by one site, but another representation of the information. When we map from the global to the local level, we can have differ-

ent transformations from the different local schemas. We represent these mappings as discussed in [15]:

$$Year_{global}(Paper, Year) \rightarrow Conference_{DBLP}(Paper, Conference), \\ Conference\_Year_{DBLP}(Conference, Year).$$

$$Title_{global}(Movie, Name) \rightarrow Title_{Amazon}(Movie, Name).$$
$$Title_{global}(Movie, Name) \rightarrow Name_{IMDB}(Movie, Name).$$

4 Metadata Extraction and Transformation

Now that we know how our global ontologies look like and how the data extracted from the web pages is structured, we have to describe how exactly we transform data from local data sources into RDF instances corresponding to the global ontologies. We need the following two steps to go from web pages to contextual metadata:

- *extract* task related metadata from distributed, inhomogeneous sources into local schemas (Section 4.1)
- *transform* this gathered metadata into one, common context schema (Section 4.2)

Even when web pages change, their structure tends to stay the same so any wrappers or transformation rules remain valid. Metadata are then extracted automatically when the web page is visited again. Of course, initial effort has to be invested for completely new wrappers or ontologies.

4.1 Extraction of Web Information Using Lixto

As described in Section 3.1, we are interested in the structured information containted in web pages. We use the Lixto Toolkit [12] for handling this extraction part. The Visual Wrapper from Lixto [2] provides a methodology and tool for the visual and interactive generation of query wrappers - programs, that automatically extract data from semi-structured data sources like web pages and transform them into XML. Lixto wrappers contain queries in the Elog query language, which is based on monadic Datalog plus extensions for regular expressions. The extractor, using as input an HTML document and a previously constructed program, generates as its output a pattern instance base, a data structure which encodes the extracted instances as hierarchically ordered trees and strings. For our publication scenario, an excerpt of the XML extraction of the example page depicted in Figure 3 looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
    <Publication>
        <Title>Super-Peer-Based Routing and Clustering Strategies for
                RDF-Based Peer-to-Peer Networks</Title>
        <Author>Wolfgang Nejdl</Author>
        <Author>Martin Wolpers</Author>
        <Author>Wolf Siberski</Author>
        <Author>Christoph Schmitz</Author>
        <Author>Mario Schlosser</Author>
        <Author>Ingo Brunkhorst</Author>
```

```
        <Author>Alexander Loeser</Author>
        <Year>2003</Year>
        <Link>http://citeseer.ist.psu.edu/..../nejdl03superpeerbased.pdf</Link>
        <Citations>
            <CitationTitle>Chord: A Scalable Peer-to-Peer Lookup Service for
                           Internet Applications</CitationTitle>
            <CitationTitle>A Scalable Content-Addressable Network</CitationTitle>
            <CitationTitle>Mediators in the Architecture of Future
                           Information Systems</CitationTitle>
            <CitationTitle>The TSIMMIS Project: Integration of Heterogeneous
                           Information Sources</CitationTitle>
            <CitationTitle>A Measurement Study of Peer-to-Peer File Sharing
                           Systems</CitationTitle>
            ........
        </Citations>
        <CitedBy>
            <CitedByTitle>Self-Organization of a Small World by Topic</CitedByTitle>
            <CitedByTitle>Semantic Query Routing and Processing in P2P Database
                          Systems: The ICS-FORTH SQPeer Middleware</CitedByTitle>
            <CitedByTitle>Top-k Query Evaluation for Schema-Based Peer-to-Peer
                           Networks</CitedByTitle>
            <CitedByTitle>Super-Peer-Based Routing Strategies for RDF-Based
                          Peer-to-Peer Networks</CitedByTitle>
            .........
        </CitedBy>
    </Publication>
</document>
```

For this XML output, a small part of the Lixto extractor, which harvests data about the title of a paper, is shown in the following snippet:

```
<StringSourceDef description="" maxInstances="-1" name="Title"
                 parentName="TitleLine">
    <ExtractionRules>
        <StringExtractionRule description="" parent="TitleLine">
            <Head description="">
                <I>
                    <Var name="0"/>
                </I>
                <O>
                    <Var name="1"/>
                </O>
            </Head>
            <AtomChain>
                <SubText description="">
                    <I>
                        <Var name="0"/>
                    </I>
                    <STD>
                        <SimpleSTD>
                            <RE pattern="(([A-Z]|[A-Z]+|[A-Z]\w+).*(\s|\-)([A-Z]\w+|[A-Z]+|[a-z]+))"/>
                        </SimpleSTD>
                    </STD>
                    <O>
                        <Var name="1"/>
                    </O>
                </SubText>
            </AtomChain>
        </StringExtractionRule>
    </ExtractionRules>
</StringSourceDef>
```

In a second step, the extracted XML data are then transformed to RDF using an XSLT script, resulting in the following format:

```
<rdf:Description rdf:about="http://citeseer.ist.psu.edu/569523.html">
    <dc:publisher>University of Hannover</dc:publisher>
    <dc:title>Super-Peer-Based Routing and Clustering Strategies for RDF-Based
              Peer-to-Peer Networks</dc:title>
    <dc:creator>
        <rdf:Seq>
              <rdf:li rdf:resource="#Wolfgang Nejdl"/>
              <rdf:li rdf:resource="#Martin Wolpers"/>
              <rdf:li rdf:resource="#Wolf Siberski"/>
              <rdf:li rdf:resource="#Christoph Schmitz"/>
              <rdf:li rdf:resource="#Mario Schlosser"/>
              <rdf:li rdf:resource="#Ingo Brunkhorst"/>
              <rdf:li rdf:resource="#Alexander Loeser"/>
        </rdf:Seq>
    </dc:creator>
    <dc:date>2003</dc:date>
    <dc:identifier>http://citeseer.ist.psu.edu/..../nejdl03superpeerbased.pdf
                  </dc:identifier>
    <citeseer:cites>Chord: A Scalable Peer-to-Peer Lookup Service for
                    Internet Applications</citeseer:cites>
    <citeseer:cites>A Scalable Content-Addressable Network
                    </citeseer:cites>
    <citeseer:cites>Mediators in the Architecture of Future Information
                    Systems</citeseer:cites>
    <citeseer:cites>The TSIMMIS Project: Integration of Heterogeneous
                    Information Sources</citeseer:cites>
    <citeseer:cites>A Measurement Study of Peer-to-Peer File Sharing
                    Systems</citeseer:cites>
    .........
    <citeseer:cited_by>Self-Organization of a Small World by Topic
                       </citeseer:cited_by>
    <citeseer:cited_by>Semantic Query Routing and Processing in P2P Database
                       Systems:The ICS-FORTH SQPeer Middleware</citeseer:cited_by>
    <citeseer:cited_by>Top-k Query Evaluation for Schema-Based Peer-to-Peer
                       Networks</citeseer:cited_by>
    <citeseer:cited_by>Super-Peer-Based Routing Strategies for RDF-Based
                       Peer-to-Peer Networks</citeseer:cited_by>
    ..........
</rdf:Description>
```

4.2 Transformation into Task Specific Semantic Views Based on the Mapping Rules

After the extraction and transformation of data according to our local schemas, we then have to transform these data into the global schema, which gives us a unified view on *all* the local sources that we can query for each scenario. The ontologies described in Section 3.2 provide these task specific semantic views, specifying the final format for the contextual metadata for each scenario.

This transformation step is facilitated by the mapping rules (see Section 3.3) which provide the translation between the local properties and relations identified on the web sites (e.g., CiteSeer local ontology) and the properties and relations that are specified in the syndicated ontology (e.g., research ontology). The mapping rules are necessary for all items we want to keep for the global view. In order to materialize our task specific semantic views, we translated the mapping rules in Section 3.3 into TRIPLE rules [18]. The following examples show how contextual information described in the research ontology and in the movie ontology can be constructed from data extracted from the local sources (DBLP, Amazon and IMDB in this case):

```
FORALL PAPER,YEAR  PAPER[global:has_published->YEAR] <-
    EXISTS CONFERENCE conference(PAPER,CONFERENCE)@DBLP.

FORALL MOVIE,NAME  MOVIE[global:has_title->NAME] <-
    title(MOVIE,NAME)@AMAZON
    OR name(MOVIE,NAME)@IMDB.
```

4.3 Triggering These Transformation Steps

Our current Beagle^{++} prototype [6, 5] is being built on top of the open source Beagle desktop search infrastructure, which we extended with additional modules (metadata generators) handling the creation of contextual information, and a ranking module, which computes the ratings of resources so that search results are shown in the order of their importance. Compared to existing desktop search applications this makes it easier to find relevant resources based on the additional contextual information and to use link-based ranking algorithms like PageRank operating on the context information graph, in addition to traditional TF/IDF measures.

Beagle Event Based Architecture. The main characteristic of our extended desktop search architecture is metadata generation and indexing on-the-fly, triggered by modification events generated upon occurrence of file system changes. Events are generated whenever a new file is copied to hard disk or stored by the web browser, when a file is deleted or modified, when a new email is read, etc, and according to the type of events, we trigger the appropriate annotation steps. Much of this basic notification functionality is provided in Linux by an inotify-enabled Linux kernel, which is used by Beagle.

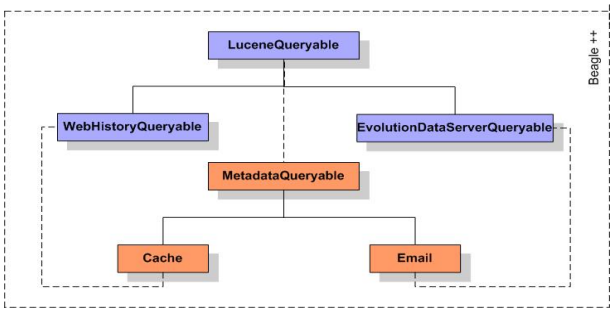


Fig. 7. Beagle Extensions for Metadata Support

Web Cache Metadata Generator. Figure 7 shows how additional metadata generators are integrated into our Beagle^{++} prototype. The queryable responsible for the web cache annotation is WebHistoryQueryable. Each URL typed in the web browser that is not in the cache will be transmitted by Beagle^{++} to the Lixto wrapper that harvests the data according to the appropriate local ontologies. The XML data are then translated

via XSLT into RDF and then are materalized into the global semantic views with the help of the TRIPLE mapping rules, stored into a RDF file, and indexed appropriately. If the URL is in the cache, the relevant metadata will be displayed.

4.4 Metadata Visualization

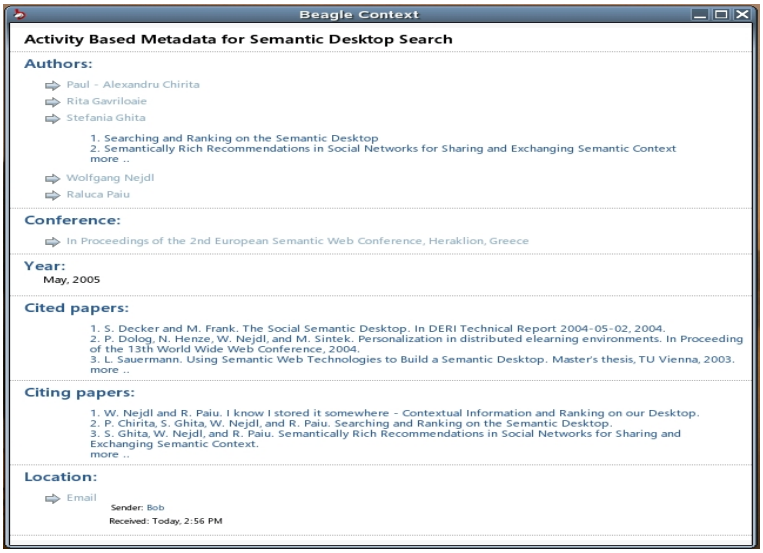


Fig. 8. Beagle^{++} Metadata Window

Let's suppose that Alice searched for the words "**semantic desktop**", and she chose among the results the paper "Activity Based Metadata for Semantic Desktop Search". When visualizing this result, the corresponding metadata can be seen as well. A new window pops up displaying a list of details that correspond to the ontology related to the type of resource. The result is stored on the desktop as a file sent as an attachment by Bob. The metadata window displays the annotations corresponding to publications together with other contextual information associated with it, retrieved from all the other sources (e.g., DBLP, CiteSeer). The publication has 5 authors and for each of the authors we can further display the next level of metadata. For example, in Figure 8, Alice extended author S. Ghita and she can see other publications of this author. Additionally, she can see that the publication was presented at the ESWC conference in 2005, its referenced publications and the ones that cited it. Information related to the provenance of this resource is also shown, the email it was saved from and its sender.

We are currently extending our prototype to be able to display metadata on arbitrary ontologies.

5 Related Work

Our approach integrates ideas from various fields: metadata extraction and syndication in the web as well as recent achievements for the semantic desktop.

In [14], the authors describe an approach for personalized content syndication, featuring a central content syndicator instance which answers user requests. Our approach differs from this approach as we do not focus on content brokerage but on metadata brokerage, and incrementally construct metadata which we use for modeling a user's context and preferences. Therefore, formats for content syndication like RSS [17] are not expressive enough to create the metadata needed. A related approach creating metadata descriptions on behalf of a web extraction process is described in [3]. The author creates RDF descriptions about publication information from dedicated sites in an automated process, as well as new views on the data based on these descriptions and additional background knowledge available for this application.

One of the most interesting semantic search efforts concerning metadata enrichment of results and their visualization is being performed in the TAP project [13]. TAP builds upon the TAPache module, which provides a platform for publishing and consuming data from the Semantic Web. Its knowledge base is updated with the aid of the onTAP system, which includes 207 HTML page templates, being able to read and extract knowledge from 38 different high quality web sites. The key idea in TAP is that for specific searches, a lot of information is available in catalogs and backend databases, but not necessarily on web pages crawled exhaustively by Google. The semantic search results are independent of the results obtained via traditional information retrieval technologies and aim to augment them. In contrast, metadata information in our scenarios reflects contextual and activity-based metadata information available on our desktop.

The difficulty of accessing information on our computers has prompted several first releases of desktop search applications during the last months. The most prominent examples include Google desktop search [11] and the Beagle open source project for Linux [9]. They do not exploit metadata information, but rely on a regular text-based index. Apple Inc. has integrated an advanced desktop search application (named *Spotlight Search* [1]) into their new operating system, Mac OS Tiger. Even though they did add semantics into their tool, only explicit information is used, such as file size, creator, last modification date, or metadata embedded into specific files (images taken with digital cameras for example). While this is indeed an improvement over regular search, it still misses contextual information often resulting or inferable from explicit user actions or additional background knowledge, as discussed in this paper.

6 Conclusions and Future Work

In this paper we discussed how relevant data can be automatically extracted from web sites visited by the user during his work and syndicated into task specific semantic views, which represent contextual information relevant for specific tasks and contexts. This contextual information can be exploited to enhance desktop search beyond full-text indexing, leading to more search results as well as to richer result representation.

We are currently putting all implementation pieces together in our Beagle++ prototype, and will evaluate it in more depth in the application scenarios described in this

paper. Additionally, we intend to investigate in more detail how to incrementally update views whenever the content of revisited web pages has changed, in order to keep our contextual information consistent.

References

1. Apple spotlight search. http://developer.apple.com/macosx/tiger/spotlight.html.
2. Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Declarative information extraction, web crawling, and recursive wrapping with lixto. In *6th International Conference on Logic Programming and Nonmonotonic Reasoning*, Vienna, Austria, 2001.
3. Robert Baumgartner, Nicola Henze, and Marcus Herzog. The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. In *ESWC*, Heraklion, Greece, May 29 - June 1 2005.
4. Andrea Cal, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the expressive power of data integration systems. In *21st Int. Conf. on Conceptual Modeling*, 2002.
5. P.-A. Chirita, S. Ghita, W. Nejdl, and R. Paiu. Semantically enhanced searching and ranking on the desktop. In *ISWC, November*, 2005.
6. Paul Alexandru Chirita, Rita Gavriloaie, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Activity based metadata for semantic desktop search. In *Proceedings of 2nd ESWC*, Heraklion, Greece, May 2005.
7. Rada Chirkova, Alon Y. Halevy, and Dan Suciu. A formal perspective on the view selection problem. In *Proceedings of the 27th International Conference VLDB*, pages 59–68, Rome, Italy, September 2001. Morgan Kaufmann.
8. Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.
9. Gnome beagle desktop search. http://www.gnome.org/projects/beagle/.
10. Cheng Hian Goh, Stéphane Bressan, Stuart Madnick, and Michael Siegel. Context interchange: new features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–270, 1999.
11. Google desktop search application. http://desktop.google.com/.
12. Georg Gottlob, Christoph Koch, Rober Baumgartner, Marcus Herzog, and Sergio Flesca. The Lixto Data Extraction Project — Back and Forth between Theorie and Practice. In *ACM Symposium on Principles of Database Systems (PODS)*, volume 23. ACM, June 2004.
13. R. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the 12th International Conference on WWW*, pages 700–709. ACM Press, 2003.
14. W. Kießling, W.-T. Balke, and M. Wagner. Personalized content syndication in a preference world. In *EnCKompass Workshop on E-Content Management*, Eindhoven, The Netherlands, 2001.
15. Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.
16. Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 95–104, San Jose, Calif., 1995.
17. RDF Site Summary specification. http://web.resource.org/rss/1.0/.
18. Triple, an rdf rule language. http://triple.semanticweb.org/.
19. Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

Towards a Semantic Wiki Experience – Desktop Integration and Interactivity in WikSAR

David Aumueller, Sören Auer

Department of Computer Science
University of Leipzig, Augustusplatz 10-11, 04103 Leipzig, Germany
{david|auer}@informatik.uni-leipzig.de

Abstract. Common Wiki systems such as MediaWiki lack semantic annotations. WikSAR (Semantic Authoring and Retrieval within a Wiki), a prototype of a semantic Wiki, offers effortless semantic authoring. Instant gratification of users is achieved by context aware means of navigation, interactive graph visualisation of the emerging ontology, as well as semantic retrieval possibilities. Embedding queries into Wiki pages creates views (as dependant collections) on the information space. Desktop integration includes accessing dates (e.g. reminders) entered in the Wiki via local calendar applications, maintaining bookmarks, and collecting web quotes within the Wiki. Approaches to reference documents on the local file system are sketched out, as well as an enhancement of the Wiki interface to suggest appropriate semantic annotations to the user.

1 Introduction

The ease of content authoring and publishing on the World Wide Web culminates in Wiki environments. The huge free encyclopaedia Wikipedia for example was created by an ever-growing community of contributing end-users. To manage the vast amount of content therein categories and lists of related concepts are manually maintained by certain users. Extending the Wiki syntax to allow for simple semantic annotations would cut down the effort of keeping such collections up-to-date, as also proposed in [5]. The WikSAR [1] prototype demonstrates a straightforward approach to enter semantics in a Wiki and presents a variety of features building on these annotations. The additional expenses in semantic authoring are enticed by instantaneously gratifying the users in various ways. Especially, the new interactive graph visualisation and navigation possibilities make the creation of meaningful connections between concepts attractive.

The next section explains the features of the semantic Wiki prototype – characteristics that common Wiki systems do not offer – i.e. how to enter semantics and take advantage of them. Section 3 explains the possibilities currently available in WikSAR to integrate the user's desktop and outlines potentials for further future approaches. Section 4 presents the interactive graphical navigation system novel to WikSAR.

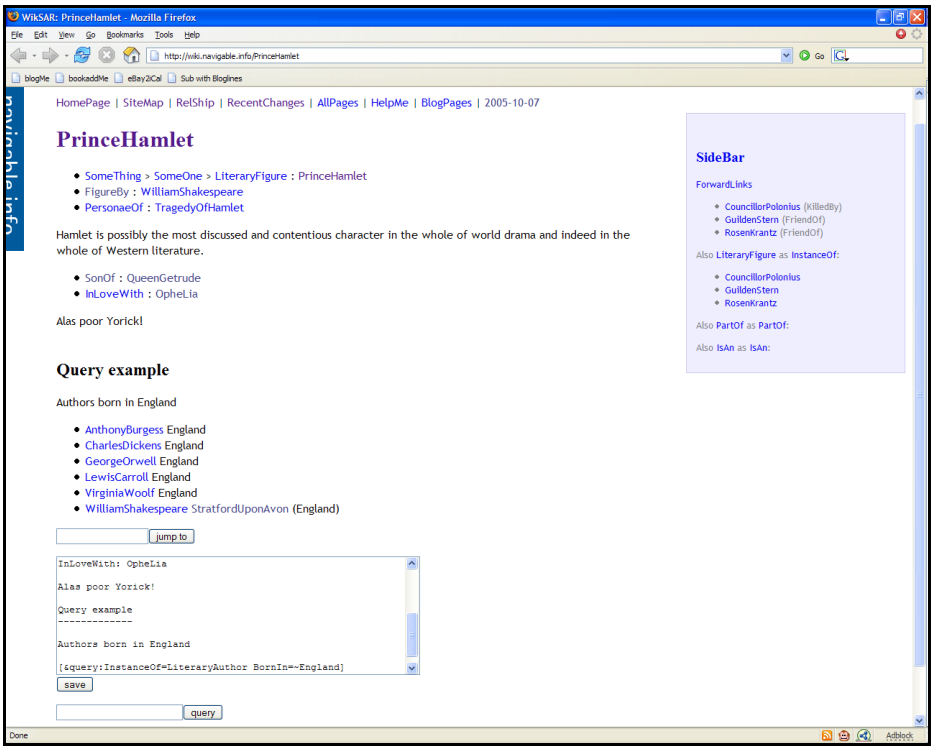


Figure 1. Interface of the WikSAR prototype.

2 Semantic Wiki features

The WikSAR interface consists of a Wiki page and the optional interactive graph visualisation that can be switched on and off within a separate frame. A Wiki page in WikSAR is divided into three parts (Figure 1): The form for editing the text, the rendered Wiki text above, and the sidebar on the right containing context-dependent links, constructed merely from semantic information present on other Wiki pages.

2.1 Semantic authoring

The WikSAR prototype uses the WikiWord or CamelCase syntax similar to Ward Cunningham's original Wiki Wiki Web [6]. However, WikiWords are not only used to create hyperlinks to other Wiki pages but are interpreted either as subject, predicate, or object in Semantic Web statements, i.e. RDF triples. The page name of a Wiki page here always denotes the subject of statements embedded in the Wiki text. Predicate and object are simply entered on an empty line within the Wiki text with WikiWords separated by a colon and space. Thus, on a page named "PrinceHamlet" the line "FigureBy: WilliamShakespeare" represents the statement <*PrinceHamlet*>

<FigureBy> <WilliamShakespeare>. By combining all such embedded statements a formal ontology emerges within the Wiki. An important aspect in ontology creation regarding the Semantic Web is to adhere to given schemas and to use already established vocabularies. An editing assistance in that direction can be integrated in the Wiki by suggesting already used WikiWords or vocabulary from external reference ontologies. The recently emerged notion of mixed client and server-side Web applications (e.g. AJAX[1] Asynchronous Javascript and XML) eases the creation of intuitively usable interfaces, such as Google Suggest[2] and del.icio.us[3] tag suggestions – these techniques can support the user in editing semantic statements in the Wiki.

2.2 Semantic navigation in context

The entered statements are used immediately to create links to related pages depending on the current context. Firstly, breadcrumbs inform the user about her position in the Wiki showing the path back to the root of the site or concept. Secondly, the sidebar shows pages or concepts related to the current concept including their type of relationship. These special *backlinks*, i.e. typed backlinks, include links to pages that contain e.g. more detailed information about mentioned concepts on the current page, often leading deeper into the site or taxonomy. E.g. on the page WilliamShakespeare there would be a typed backlink that leads to "PrinceHamlet" which is a "FigureBy" the current concept. The breadcrumbs on the other hand show the way back, up the hierarchy, saying e.g. that WilliamShakespeare is an author, which again is a person. Thus, two important questions regarding the usability of a web site get answered: "Where am I?" and "Where can I go?". In addition to these navigational means on every page, the semantic annotations are used to automatically generate a class hierarchy or a complete map of the ontology as labeled graph, i.e. a typed site map.

2.3 Semantic retrieval

The triples created in WikSAR are available for semantic queries. Using a triple store that supports RDF query languages, such as RDQL and SPARQL, the Wiki space can be queried to return distinct concepts (pages). The current query syntax in WikSAR allows filtering by specific predicate-object combinations, as in "predicate operator object", allowing equality, quantitative comparisons, and regular expressions as operator. Concatenating expressions by spaces implies 'and'-semantics. The query "InstanceOf=LiteraryAuthor BornIn=~England DateOfBirth between 1800 and 1900" returns a list of authors born in the 19th century in England, as demonstrated in [1].

2.4 Semantic views and query chaining

WikSAR accepts a variety of proprietary commands embeddable in Wiki pages to generate and include content gathered from all available data. For example the links

[1] http://www.adaptivepath.com/publications/essays/archives/000385.php
[2] http://www.google.com/webhp?complete=1
[3] http://del.icio.us

in the sidebar get created by putting special operators in the template Wiki page for the SideBar. Likewise, queries in WikSAR need not produce only search results available temporarily to the querying user; queries can be embedded persistently within Wiki pages, too. Whenever a page containing a query statement gets loaded the specified query is executed and the result is embedded seamlessly within the Wiki page, thus creating a special view on the Wiki information space. That way it is possible to maintain pages containing always up-to-date query results, e.g. collections of related concepts such as EnglishAuthors or StrongBeer. Newly entered content anywhere on the Wiki will immediately be reflected in dependant collections or concepts as well – without any manual effort. We plan to use such collections further to create more complex queries by allowing to chain queries, i.e. by using the result set of one query as input for another one. Allowing any list of concepts as input (even manually created lists, e.g. lists only of personal interest) would facilitate retrieval possibilities for very specific interest. In a Wiki collecting e.g. scientific publications, a user might have lists of favourite authors and favourite subjects; using these as input to a query she could retrieve publications of her explicit needs and have the list always available as distinct Wiki page. Publishing query results or single Wiki pages as RSS-feed informs the user immediately about changes matching her special interest without the need to visit the Wiki.

3 Desktop Integration

Due to their flexibility Wikis are often used for Personal Information Management (PIM), as already suggested by [6]. Important in this context is to bridge the gap between the Wiki accessible only by a web browser and the local desktop environment, i.e. local applications and the file system. Both worlds have their advantages for PIM; the Wiki and the information therein residing on a web server are omnipresent, accessible from any machine with Web access. Conversely, reminders of appointments can be better triggered locally by calendar applications, and collections of user-centric files, such as office documents and PDFs, usually reside locally, as well.

WikSAR publishes dates entered in the Wiki as remote calendar entries in the iCalendar format. Such calendar files accessible by the HTTP protocol can be imported or subscribed to by desktop calendar applications (e.g. Mozilla Sunbird), which can then for example trigger reminders.

Interesting PIM applications are the maintenance of lists of web bookmarks, as well as of quotes from web pages. WikSAR uses so-called bookmarklets (some JavaScript statements that are called from within a web browser by a single user click to provide access to information on the currently open web page) to put selected pieces of text and/or the URI of the resource onto the Wiki. Such collections may be of interest to single person or a whole community. Using a Wiki to maintain bookmarks and web quotes makes them accessible not only from the web browser where they got collected but from anywhere. Further, they can be easily annotated, too.

Accessing the local file system from within a remote web application is impossible due to security reasons. We plan to be able to reference local files from within the Wiki nevertheless by either providing a link to one's desktop search engine as these

search engines (e.g. Google Desktop Search) are more and more accessible via HTTP calls on *localhost*. Another approach may utilize the magnet URI scheme, an open scheme "enabling seamless integration between websites and locally-running utilities"[4]. This is already successfully applied in the P2P-domain by providing unique keys for files that file-sharing applications can then search for. To reference emails of a PIM user, cross platform component models to email clients may provide the missing link, such as XPCOM[5] for Mozilla Thunderbird.

4 Interactive Graph Visualisation and Navigation

Visualisation of complex information spaces is an important area of research in itself to help understand data, e.g. [2]. In the Wiki context the link structure, i.e. the interconnections between Wiki pages, may be drawn as a graph, as e.g. done by TG WikiBrowser[6]. Labelled graphs as available in WikSAR via the entered Semantic Web statements offer more sophisticated visualisation and filter possibilities as merely a graph of un-typed hyperlinked pages. Visualising ontologies is possible for example in Protégé[7]. WikSAR offers a novel approach by integrating an interactive graphical representation within the Wiki; navigating through the Wiki space is possible in either the Wiki or the graph, changing focus simultaneously in both views.

The graph of WikSAR pages is built upon the entered semantic triples. These are interpreted as directed labelled graph with subjects and objects being nodes, and predicates becoming labelled edges. Since an object of one triple may be the subject of another, these edges are chained to spawn a directed graph. To overcome the problems of visualising large graphs, the popular fisheye views and hyperbolic trees show the part of the graph of interest more prominently than the rest of the graph.

For the Wiki we chose a special tree layout algorithm that always renders the concept in focus as root of a tree and grows the tree via breadth-first-search along the graph, as provided by the prefuse toolkit [4] as radial tree layout[8]. Here, the node in focus, i.e. the root, resides in the centre of the visualisation, with the children attached. Deeper descendants get displayed further away from the centre such that every level of the hierarchy resides on its own circle around the root. For the Wiki context this tree layout got adapted to display less information the deeper the concept resides in the current tree, i.e. the concept in focus and its immediate children are displayed with labels both for concept nodes and edge type, whereas deeper nodes merely are cherished with a circle instead of label and without labelling the edge type. By default, concepts being further away than four hops from the current one in focus do not show up at all. The visualisation will be enhanced by allowing the user to control the displayed tree depth and filter or highlight specific types of edges.

[4] http://magnet-uri.sourceforge.net
[5] http://xulplanet.com/references/xpcomref/group_Mail.html
[6] http://www.touchgraph.com
[7] http://protege.stanford.edu
[8] Due to patent restrictions the hyperbolic tree layout algorithm is not available in prefuse.

We believe that the interactive graph helps to understand the relationships between concepts in WikSAR by providing a playful approach to examine the information space, e.g. the relationships among the characters in a Shakespearian play (Figure 2).

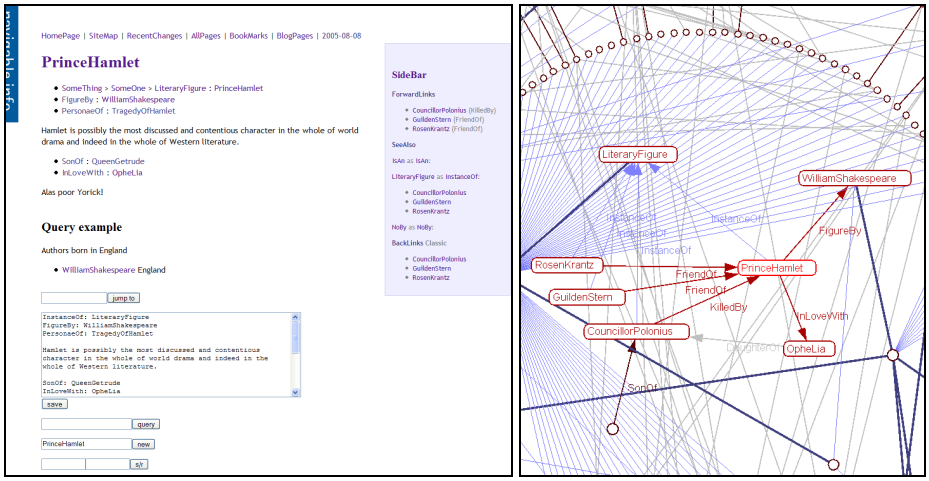


Figure 2. Interface of the WikSAR prototype with interactive graph.

5 Conclusion

With WikSAR, the presented Wiki prototype, it is straightforward to author semantically annotated content and to enjoy the benefits thereof, a challenge identified by [3]. A simple query interface offers retrieving information not only by keyword but also with respect to specific semantics represented in the Wiki. Persistently embedding queries into Wiki pages allows to create automatically updated collections of concepts and monitoring of specific changes of content collected on the site. Browsing the Wiki ontology as interactive graph representation further enhances navigation and helps to understand complex structures by giving a graspable overview. Future enhancements to bridge the gap between the Wiki application on the Web and local applications will improve personal information management.

References

[1] Aumüller, D., Semantic authoring and retrieval within a Wiki. In Demo Session at the ESWC 2005, Heraklion, Greece (2005). Available at <http://wiksar.sf.net>.
[2] Geroimenko, V., Chen, C. (Eds.) Visualizing the Semantic Web. Springer (2003)
[3] Halevy, A. Y., *et.al.*: Crossing the Structure Chasm. Conf. on Innovative Data Systems Research, California (2003)
[4] Heer, J., Card, S. K., Landay, J. A.: prefuse: a toolkit for interactive information visualization. In CHI 2005, Human Factors in Computing Systems (2005)
[5] Krötzsch, M., Vrandečić, D., Völkel, M.: Wikipedia and the Semantic Web – The Missing Links. In Wikimania 2005, 1st Int. Wikimedia Conf., Frankfurt, Germany (2005)
[6] Leuf, B., Cunningham, W.: The Wiki way: quick collaboration on the web. AW (2001)

An activity based data model for desktop querying (Extended Abstract)*

Sibel Adalı[1] and Maria Luisa Sapino[2]

[1] Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA,
sibel@cs.rpi.edu,
[2] Università di Torino, Corso Svizzera, 185, I-10149 Torino, Italy
mlsapino@di.unito.it

1 Introduction

With the introduction of a variety of desktop search systems by popular search engines as well as the Mac OS operating system, it is now possible to conduct keyword search across many types of documents. However, this type of search only helps the users locate a very specific piece of information that they are looking for. Furthermore, it is possible to locate this information only if the document contains some keywords and the user remembers the appropriate keywords. There are many cases where this may not be true especially for searches involving multimedia documents. However, a personal computer contains a rich set of associations that link files together. We argue that these associations can be used easily to answer more complex queries. For example, most files will have temporal and spatial information. Hence, files created at the same time or place may have relationships to each other. Similarly, files in the same directory or people addressed in the same email may be related to each other in some way. Furthermore, we can define a structure called "activities" that makes use of these associations to help user accomplish more complicated information needs. Intuitively, we argue that a person uses a personal computer to store information relevant to various activities she or he is involved in. Files may be related to activities either directly or indirectly with some degree of relationship. In this paper, we define a simple model of an activity and show the types of queries that can be answered using the activity model. Our model assumes that activities can involve files that are related to each other in many different ways: a period of time that may contain disjoint intervals, different locations, a group of people that we interact with and various combination of these types of associations. Furthermore, files may be related to multiple activities independent of their participation in one activity. Finally, our model aims to find the best indicators of an activity for a specific user and computer based on the data provided by that user.

* This work was supported by the National Science Foundation under grants EIA-0091505 and IIS-9876932.

2 Activity based querying

As a movitating example, suppose the user wants to find the photo of the Panda from her trip to the zoo and her photos do not have the necessary tags. It is possible to search for this information by first finding the time frame for the specific trip to the zoo by using a keyword query for all the relevant files and then limit the search to files created or photos taken at this time frame. Similarly, it is possible to limit searches to relevant people, directories based on the user's needs and find information by following associations known to her. In this case, we are able to find specific information and at the same time follow the links to browse the related information along different dimensions. This is similar to the way we recall information that we do not remember. To accomplish this, the system simply needs to show the relevant associations for any searched query.

To facilitate this type of querying, we define the notion of an activity as follows: Suppose $\mathcal{O}$ refers to the universe of objects that could be stored in the computer. Then, an *activity* actF is defined as a function $\mathsf{actF}_{\preceq} : \mathcal{O} \rightarrow D_\tau$ where $\tau = (D_\tau, \preceq)$ is any partial order. Intuitively, an activity is an outside event that triggers the use of a computer and the creation or use of data. Examples of professional activities that an academician may be involved in are publishing papers at conferences or journals, sending proposals, teaching classes, etc. Examples of personal activities may be taking trips, participating in sportive activities and personal gatherings, etc. We are not interested in modeling the meaning of these activities, but how they cause the creation of data objects for this specific user. For example, for a trip to visit friends or family, pictures taken at that trip, emails and web site visits corresponding to purchase of tickets and email correspondence with friends can all be considered relevant to the trip. These in fact model different aspects of the trip. For a conference, we might also create documents such as papers and presentations in addition to the files associated with a trip. To define an activity, we assume the user defines an *activity schema* actS as an ordered list $\mathsf{actS} = \langle\, lf_1 \ldots lf_k \,\rangle$ of logical formulae lf_i constructed from predicates defining the "where", "when", "what" type of constraints with possible crisp or fuzzy semantics. The activity actF defined by the above schema is then given by:

$$\mathsf{actF}(o) = \begin{cases} min\{i \mid o \models lf_i\} & \text{if } \exists i.(1 \leq i \leq k) \wedge o \models lf_i \\ k+1 & \text{otherwise} \end{cases}$$

for any object $o \in \mathcal{O}$. The ordering of constraints gives further information about the ordering of relevance where each object belongs to the highest priority logical formula that is satisfied by the properties of the object. For fuzzy constraints, we assume the existence of fuzzy logical operators and functions that merge sorted lists containing objects and scores.

To further enhance the functionality of the system, we develop clustering methods to find the common properties of objects for an activity. The aim is to help the user by showing relevant properties of objects for an activity beyond those that are specified by the user. Being able to identify and sort files in

relationship to an activity and find the most relevant properties of objects for an activity allows us to perform the following set of tasks on top of the enhanced search queries that we discussed earlier:

- *Show me the files on the visit to Company Acme last year.* Find the dates, people involved in the visit, files created for the trip and organize them in the order of relevance together with the relevant categories of information.
- *Organize my emails based on the known activities.* Parse important properties for each activity and place each mail in one or more activities based on how well they match the given activity (how many properties it matches).
- *Limit my keyword search to those items relevant to activity "Writing the activity paper".* Order the matching items with respect to their match to the given activity.
- *Hide all items relevant to activity "Car Purchase" in all my searches.* Given a level of sensitivity, do not show the items that appear to be related to a specific activity. For example, in a professional setting, do not show files related to personal use of the same computer. This allows the user to implement their own notion of privacy in different settings.
- *Order all files based on their relationship to this file.* Given a video clip, we can find other related items such as presentations we have given with that video clip or the people we met during these meetings. We can also limit the search to a specific activity to focus the search further.
- *Show me all related activities for a specific time/person/place.* If a number of activities are known to the computer, than we can search and find out which activities we were involved in a specific period of time or a given place. This allows us to recall "history" as it is relevant to us.

We are currently working on a prototype of our system to illustrate the above mentioned functionality.

3 Related Work

When the available information is stored on the users' desktops, it is important for information management applications to be able to model users' interpretation of their data and to capture the possibly different meanings, semantics links, and relationships that the users associate to the information units available. For this purpose, various Personal Information Management tools are being developed to assist the user with her navigation/browsing over various forms of personal digital data [10, 5, 4, 8, 13, 12].

MyLifeBits [10] is a research project and a software environment which aims at storing, in digital form, *everything* related to the activities of an individual and providing full-text search, text and media annotations, and hyperlinks to personal data. Another Microsoft project, *Stuff I've Seen* [5], aims at managing personal data, such as already-read email messages, for reuse. Retrieval and presentation of information are based on contextual cues, such as time and author in the case of email.

Recently, there is more work on personal desktop information management. Chandler [4], for instance, is an interesting open source example of such management tools, integrating calendar, email, contact management, task management, notes, and instant messaging functions. Haystack [8] and Gnowsis [13, 12] are systems that adopt the semantic web data modeling approach, and treat all the data objects stored on the desktop as resources on which semantic networks are defined using the Web Consortium's *Resource Description Framework (RDF)* [11].

More user centered treatment of object semantics recently lead to a new emerging research area referred to as *Experiential Computing* [6, 2, 1]. According to this approach, the user interaction systems should exploit and reflect as closely as possible users' previous experiences. Thus, users should be part of the complete system. Experiential environments allow a user to directly observe data and information of interest related to an event and to interact with the data based on his or her own interests in the context of that event. By developing experiential environments, researchers aim to develop new generation information management systems which transform database applications from being simply information sources to being powerful insight and experience sources. The data generated for each event is experienced by an observer and interpreted to create knowledge. In this knowledge production process, the observer plays an important role to interpret the data, and capture the experienced semantics. Recently, there is interest in developing methods to exploit relationships between objects for data cleaning problems [7].

Our approach differentiates from all of the above systems. Based on the fact that objects in a desktop may be related to each other in different ways in different contexts, we argue that users create and modify data as a function of activities that they are involved in. The relatedness of an object to an activity is a fuzzy notion. We develop methods to define and query activities. This allows users to not only locate relevant information but also organize their desktop in relationship to these activities.

4 Conclusions and Future Work

Our notion of an activity - a way to group objects in a user's desktop into overlapping clusters of related objects and related properties - is a first step towards solving the problem of scale when dealing with an ever increasing amount of data both on our own desktop as well as in other data sources that we use and share. Even though available semantic information such as free text or semantic annotations can be consumed easily in any desktop system including ours, generating this information is still very resource intensive. Similarly, content-based retrieval methods for image, video and other media suffer from the problem of being too general. The content of an image may be described very differently based on context. Hence, there is a need to integrate these methods with other data organization methods such as activities to facilitate their effective use.

We are in the process of implementing our prototype activity search and browse system as described in this paper. To this end, we are investigating various algorithmic and system issues in the implementation of this system. One of the main future problems we need to address is the issue of structured activities where an activity may be described by combining simpler activities. An activity may have many different aspects, for example a trip has a preparation phase, the actual trip followed by the other related activities. Based on our queries, we might be interested in a certain aspect of a given activity and the system should immediately adapt to this using a form of relevance feedback. Even though we can keep activity definitions fairly simple, we can learn about user's specific preferences based on their interactions with the system and integrate these back into the system. Our long term goal is to augment the desktop with inference tools that make use of the semantic data available in the activities to automatically associate semantics with data objects. The availability of these solutions would be an important first step towards solving the problem of scale in information systems.

Acknowledgment. We would like to thank Ramesh Jain for stimulating discussions on multimedia querying and experiential computing.

References

1. P Appan, H. Sundaram, D. Birchfield, "Communicating everyday experiences" *Proceedings of the 1st ACM workshop on Story representation, mechanism and context*, 2004.
2. S. Boll , U. Westermann, "Mediaether: an event space for context-aware multimedia experiences", *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, 2003.
3. Jan Chomicki: Preference formulas in relational queries. ACM Trans. Database Syst. 28(4): 427-466 (2003).
4. "Vision of Chandler", *www.osafoundation.org*, 2005
5. S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. "Stuff i've seen: A system for personal information retrieval and re-use." *Proceedings of SIGIR*, 2003.
6. R. Jain. "Experiential computing", Commun. ACM, vol.46(7), 2003, pp. 48–55.
7. D. V. Kalashnikov, S. Mehrotra, Z. Chen: "Exploiting Relationships for Domain-Independent Data Cleaning." SDM 2005.
8. D.R. Karger, K. Bakshi, D. Huynh, D. Quan, V. Sinha: "Haystack: A General Purpose Information Management Tool for End Users of Semistructured Data." *Proc. CIDR 2005.*
9. F. Manola and E. Miller: "RDF primer". *www.w3.org/TR/rdf-primer/*, 2003.
10. "MyLifeBits Project", *research.microsoft.com/barc/mediapresence/MyLifeBits.aspx*, 2005.
11. "Resource Description Framework (RDF)" *//www.w3.org/RDF/*, 2005.
12. L. Sauermann: "The Semantic Desktop - a basis for Personal Knowledge Management." *Proc. I-KNOW 05.*
13. L. Sauermann: "The Gnowsis Semantic Desktop for Information Integration" *Proceedings of IOA Workshop of the WM2005 Conference.*

OntoPIM: How to Rely on a Personal Ontology for Personal Information Management

Vivi Katifori[2], Antonella Poggi[1], Monica Scannapieco[1],
Tiziana Catarci[1], and Yannis Ioannidis[2]

[1] Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza" - Via Salaria 113, I-00198 Roma, Italy
surname@dis.uniroma1.it
[2] Department of Informatics and Telecommunications
University of Athens - 157-84, Ilisia, Athens, Hellas, Greece
vivi@mm.di.uoa.gr, yannis@di.uoa.gr

1 Introduction

Nowadays, our personal computer contains a huge amount of information, that is stored in several different formats, including emails, pictures, text documents, media file, address books, etc. When we need to look for some information, one possibility is to use a keyword-based search tool, such as Google Desktop [1]. We then get several links to documents, mails, databases, etc. that relate to our search but are often too scattered in order to let us easily obtain the information we are looking for, even if this information is actually contained in our desktop.

In this paper, we propose a framework for Personal Information Management (PIM), called *OntoPIM*, that relies on the use of a *Personal Ontology*, that describes user's domain of interest in terms of objects, classes and relations. The ontology is personal in the sense that it reflects the user view of her domain(s) of interest. It is used to assign a semantics to the information contained in the user desktop, as well as to query the system in order to obtain a certain information. Then, by relying on the Personal Ontology, our framework overcomes the limitations of desktop search tools available nowadays. In particular, by the use of the *Semantic Save*, it provides the user the possibility to store any object of interest according to its semantics, i.e. to relate it to the concepts of the Personal Ontology, where an object may be a mail, a document, a picture, or any other type of data. Then, the user is able to query the Personal Ontology, whereas the system carries out the task of suitably processing the query, accessing the different pieces of information involved in the query, and assembling the data into the final answer.

The main contributions of this work are therefore (i) the framework definition for Personal Information Management using a Personal Ontology, and (ii) the architecture for the system, that encompasses heterogeneous data wrapping, data integration and personalization tools. This work is part of a wider project called TIM - Task-centered Information Management - under development in the frame of the DELOS NoE [2]. TIM has the two main goals of (i) classifying personal information by means of a user-tailored ontology, and (ii) allowing task-oriented interaction with one's own PC. In this paper, we focus on the first goal.

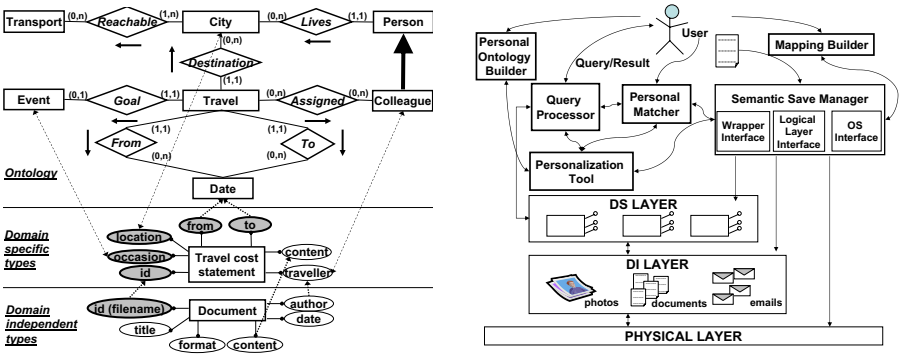
The paper is organised as follows. In Section 2, we illustrate the use of the Semantic Save. In Section 3, we discuss the architecture of the system. Then, in Section 4, we present the formal framework underlying the OntoPIM system.

2 Semantic Save

In this section, we illustrate how the OntoPIM *Semantic Save* works. Suppose that we have filled our last travel cost statement. We then proceed as follows.

- First, we indicate that we are saving an object of type **document**. The system extracts from the document a set of metadata, e.g. the **author** and the **date**. The objects that are created in this step are called *domain independent (DI) objects*, since they may exist in every domain and have always the same set of attributes.
- Second, we specialize the type of the data with respect to a particular domain. In our scenario, we indicate that the document we are saving is an object of type **travel cost statement (TCS)**, that is one of the *domain specific (DS) types* that are associated with the **business domain**. Thus, a new DS object of type **TCS** is created, whose part of the attributes is automatically mapped from part of the attributes of the DI object. This is the case of the attribute **traveller** in our example. We then may be asked to enter some other attributes associated with the **travel cost statement** DS type, as for example the **location** and the **occasion** of the travel.
- Finally, the system maps the attributes of interest of the newly created object of type **TCS** to concepts of the Personal Ontology. Note that this step is performed automatically, thanks to a set of rules, called *mappings*, that characterize each DS type and are specified when the DS type is newly created. The semantics of these mappings is that each attribute value becomes a *representation* of an instance of the concept to which the attribute is mapped. In our scenario, OntoPIM maps the attribute **traveller** of the travel cost statement to the concept **colleague**. Similarly, it maps the **location** and the **occasion** respectively to the concepts **city** and **event**.

The result of the performed Semantic Save is graphically represented in Fig. 1(a), where the ontology is represented in the flavor of a simplified Entity-Relationship model.



(a) Semantic Save result (b) OntoPIM architecture

Fig. 1.

3 The OntoPIM Architecture

The OntoPIM architecture is shown in Figure 1(b). Note that all the modules interact with three diffent data layers that, starting from the bottom, are: (i) the physical layer, storing files or relational tables or any other physical objects that can be stored on a PC;

(ii) the first wrapper layer (DI Layer) representing domain independent (DI) objects from the physical layer, such as emails, documents, photos etc., and (iii) the second wrapper layer (DS Layer) representing domain specific (DS) objects that correspond to domain specific types, such as the travel cost statement of the running example.

In what follows we describe the main OntoPIM modules.

- The user interacts with the **Personal Ontology Builder (POB)** in order to build her own Personal Ontology. Such representation is intended to be completely independent of the physical representation of information.
- The **Personalization Tool (PT)** interacts with the POB, to automate the creation and the modification of the ontology on the basis of an appropriate user profile. Moreover, the PT is responsible for automating the Semantic Save function to some extent, proposing itself possible concepts to be associated with the document, completing queries with things implied by the user, etc.
- The **Mapping Builder (MB)** allows the user to create and modify her DS types. By interacting with the user, it establishes the correspondence between DS objects of the DS Layer and concepts of the Personal Ontology. This specification is then translated into the set of rules that constitutes the set of *mappings* that will be formally introduced in the next section.
- The **Semantic Save Manager (SSM)** takes as input a physical object **o** and uses the mapping created by the MB module to perform the Semantic Save by: (i) invoking the operating system in order to save **o** in the OS file system, (ii) creating the DI abstraction of **o** and (iii) linking it to the corresponding wrapper.
- The **Personal Matcher (PM)** performs instance matching. It is responsible for identifying attribute values of different DS objects as representing the same real world entity. It produces as output the set of *matching rules* that describe how to perform the matching. These rules will be formally presented in the next section.
- The **Query Processor (QP)** is responsible to process and answer the queries posed by the user over the Personal Ontology. More specifically, the QP exploits the abstraction created by the SSM, the mapping created by the MB and the rules produced by PM, in order to rewrite the query in terms of queries to wrappers, that retrieve the actual data from the physical layer.

4 Formal Framework

In this section, we introduce the formal framework underlying the OntoPIM system, that encompasses two main functions that are the Semantic Wrapping and the Semantic Integration. The former aims at overcoming the personal data heterogeneity and its primitive lack of semantics by presenting the information contained in its mails, documents, etc. as data tuples of relations that are meaningful with respect to the user's domain of interest. On the other hand, the Semantic Integration function lets the user query the ontology, that represents its personal, integrated view of its domain of interest, while the system carries out the task of suitably retrieving, reconciling and assembling the actual data. Because of lack of space we will focus here on the more challenging part of the system, i.e. the Semantic Integration. In particular, this makes use of a simple description logic, called DL-Lite [4], to describe the Personal Ontology provided to the user. DL-Lite is tailored to capture basic ontology languages and it is particularly

suitable in our context, where the user may want to pose complex queries over a huge amount of data. Thus, in DL-Lite, answering conjunctive queries posed over the Personal Ontology can be done in polynomial time in the size of the personal data. Notably, DL-Lite comes with a system, called QUONTO [3], upon which OntoPIM is built.

Given an appropriate Semantic Wrapping layer that presents user's own data as DS objects, the Semantic Integration part of OntoPIM can be characterized by means of a quadruple $\mathcal{SI} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}, \mathcal{R} \rangle$, such that:

- $\mathcal{O}$ is the Personal Ontology, described by means of a DL-Lite TBox.
- $\mathcal{S}$ is a set of DS types.
- $\mathcal{M}$ is a set of mappings, i.e. a set of rules of the form:
  $$R_S(\boldsymbol{v}) \rightarrow conj(\boldsymbol{x}, \boldsymbol{y}), I(\boldsymbol{x}, \boldsymbol{v}),$$
  where $R_S \in \mathcal{S}$, $\boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}$ denote variables $v_1, ...v_n, x_1, ...x_n, y_1, ...y_m$, n is the arity of R_S, $m \geq 1$ and $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form $C(z)$ or $R(z_1, z_2)$, where C and R are resp. a basic concept and a role in $\mathcal{O}$, z, z_1, z_2 are variables in $\boldsymbol{x}, \boldsymbol{y}$ and $I(\boldsymbol{x}, \boldsymbol{v})$ is a set of atoms of the form $I(x, v)$ that indicates that v is a representation of the instance x. We call I *Instance relation*.
- $\mathcal{R}$ is a set of rules, called *matching rules*, that specify how to identify and match different representations of the same instance of a given concept. These rules are applied to the set of atoms generated by the mappings.They may have one of the following forms:
  1. $C(x_1) \wedge C(x_2) \wedge I(x_1, v) \wedge I(x_2, v) \rightarrow x_1 = x_2$;
  2. $C(x_1) \wedge C(x_2) \wedge I(x_1, v_1) \wedge I(x_2, v_2) \wedge sim(v_1, v_2) \rightarrow x_1 = x_2$.

  where x, x_1, x_2 are variables in $\boldsymbol{x}$, v, v_1, v_2 are variables denoting data values, C is a basic concept of $\mathcal{O}$, $sim(v_1, v_2)$ is a predicate that checks whether v_1, v_2 are similar according to a certain similarity definition, and $conj(\boldsymbol{x})$ and $I(x_i, v_i)$, are defined as above for $i = 1, 2$.

To illustrate the scenario above, let us come back to the example of the Section 2. We establish a connection between the data of interest contained in each object of type **TCS** and the Personal Ontology graphically represented in Figure 1(a) by means of the following mapping assertion:

$$\begin{aligned}\textbf{TCS}(v_1, v_2, v_3, v_4, v_5, v_6, v_7) \rightarrow \text{Goal}(x_1, x_4), I(x_4, v_4), \text{Destination}(x_1, x_3),\\ I(x_3, v_3), \text{Assigned}(x_1, x_2), I(x_2, v_2), \text{From}(x_1, x_5),\\ I(x_5, v_5), \text{To}(y, x_6), I(x_6, v_6).\end{aligned}$$

Then for each concept of $\mathcal{O}$ we define a matching rule of type 1. We also define the following matching rule of type 2 stating that two dates that are expressed in a different format represent the same instance of the concept Date:

$$\text{Date}(x_1), I(x_1, v_1), \text{Date}(x_2), I(x_2, v_2), \text{sameDate}(v_1, v_2) \rightarrow x_1 = x_2,$$

where we assume that the system is able to evaluate the predicate $sameDate(x_1, x_2)$.

Now, suppose that we are saving a travel cost statement concerning the travel that Mr. Cabernet made to participate to the World Wine Event (WWE) in Bordeaux from the 1/09/2003 to the 5/09/2003. The $\mathbf{TCS}$ mapping generates the following set of facts, that constitutes a portion of the DL-Lite ABox:

$$\begin{aligned}&\text{Travel}(x_1), \text{Event}(x_2), \text{Goal}(x_1, x_2), \text{City}(x_3), \text{Destination}(x_1, x_3),\\ &\text{Colleague}(x_4), \text{Assigned}(x_1, x_4), \text{Date}(x_5), \text{From}(x_1, x_5), \text{Date}(x_6), \text{To}(x_1, x_6).\end{aligned}$$

Moreover, the mapping generates the following portion of the Instance relation I:

Constant	Representation	Constant	Representation	Constant	Representation
x_2	WWE	x_4	Mr.Cabernet	x_6	05/09/03
x_3	Bordeaux	x_5	01/09/03		

Then, given the DL-Lite TBox expressed by means of the Personal Ontology $\mathcal{O}$, the DL-Lite ABox obtained above, the Instance relation I and the matching rules $\mathcal{R}$, the system can answer any conjunctive query over $\mathcal{O}$ and, for every constant x_i possibly returned, it proposes the set of corresponding representations, according to the computed extension of the relation I. Note that x_1 has not any representation. This is not surprising since the instances of the concept Travel would never be mapped to any attribute value. Similarly, it would not make sense to ask for an instance of the concept Travel.

5 Conclusion

We have presented a novel approach to Personal Information Management that takes advantage of the use of a Personal Ontology to store the data of one's desktop and to provide the user for an intelligent and efficient way of querying such data. We have proposed a framework that (i) overcomes data heterogeneity and lack of semantics by the use of a Semantic Wrapping function, (ii) integrates data and makes it accessible through a unified, user's conceptual view, by the use of a Semantic Integration function. Finally, we have presented the architecture of the system.

Currently, we are facing the *instance matching* problem by incorporating in the framework a set of rules responsible for detecting different representations of the same instance. In the future, we plan to investigate how to produce this set of rules. Moreover, note that once the matching rules have been applied, we actually keep all different representations of the same instance. However, sometimes we may want to correct some them. Suppose for example that in our domain view, a colleague has a unique email address. On one hand, two different email addresses may be associated to the same colleague because of spelling errors, in which case we would like to keep only the correct address. On the other hand, whenever a colleague moves, we may want to update his address while keeping the old one, in order to be able to retrieve, for example, an email that he sent us before moving. We plan to work on this in the future. Finally, we have discussed how personalization would come into play in order to help the user expressing queries, saving documents, etc. This will be the object of future deeper joint research activities. Moreover, we aim at studying how to rely on OntoPIM in order to develop a task-centered tool that would for example automatically fill a travel cost statement thanks to the data in our desktop.

References

1. Google Desktop, `http://desktop.google.com/`.
2. DELOS NoE, `http://http://delos-noe.iei.pi.cnr.it/`.
3. A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: Querying Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, 2005.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, 2005.

A Web Information Retrieval System Architecture Based on Semantic MyPortal

Haibo Yu[1], Tsunenori Mine[2], and Makoto Amamiya[2]

Department of Intelligent Systems, {Graduate School[1], Faculty[2]} of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-koen, Kasuga, Fukuoka 816-8580, JAPAN
{yu, mine, amamiya}@al.is.kyushu-u.ac.jp

Abstract. In this paper, we mainly focus on a communication mechanism which enables efficient information publishing and sharing among semantic desktops. We propose MyPortal as a "one stop" for all the information relevant to the user and further propose the conceptual architecture of a P2P community Web information retrieval system based on MyPortal. This architecture enables not only precise location of MyPortal instances and their Web resources but also the automatic or semi-automatic integration of hybrid semantic information delivered through Web content and Web services, and it also ensures that the semantics will not be lost during any part of the lifecycle of the information retrieval process.

1 Introduction

Current Web design targets human consumption, based on keywords for information indexing and searching, which not only gives rise to an enormous number of irrelevant search responses, but is unsuitable for machine processing. In addition, the user's desktop information and the published Web information are managed separately, giving rise not only to a redundancy of information but also creating difficulties in managing the relationship among items of information and applying user personalization.

Currently, there are some research projects, such as Haystack [1] and Gnowsis [3] trying to use semantic Web technology for the management of user personal desktop information. However, they lack the functionality for searching, accessing, aggregating and processing of the Web information on the fly when necessary and a unified interface for managing not only the personal desktop information but also the relevant Web information. And a reasonable architecture and efficient mechanisms for the connecting, discovering, and sharing of the information among semantic information nodes are necessary.

In this paper, we make our main concern on how to connect these information nodes in a robust and efficient way, how to discover and share the information among these information nodes and what functionalities need to be provided in order to realize these targets.

We propose our semantic Web information retrieval system architecture based on the following main ideas.

First, "combining Web portal technology with semantic desktop technology to provide a "one stop" for the user to all his relevant information." As semantic desktop provides a good solution for managing user personal information but lacks the functionality to search, collect and aggregate information from the Web for the user on the fly. On the other hand, Web portals provide a good solution for collecting relevant information for the user, but lack options for personalization and suffer from the problems of centralized architecture. We make use of the basic mechanisms for semantic personal information management of current semantic desktops and enhance their Web information publishing and sharing functionalities to construct a semantic MyPortal.

Second, "using peer-to-peer computing architecture to connect MyPortals with emphasis on an efficient method for reducing communication load." Decentralized P2P systems are robust, scalable and cheap to maintain, but tend to have large amounts of information transferred among many peers. Hence, an efficient mechanism for reducing communication loads with least loss of precision and recall is very important in a P2P information retrieval system. We propose our Agent-Community-based Peer-to-Peer information retrieval method called ACP2P to connect and manage the communication among MyPortals.

Third, "ensure that the semantics are not lost sight of during any part of the lifecycle of information retrieval." In order to enable consumer re-using semantic data, we designed the interfaces and the protocols involved in the whole life cycle of information retrieval tasks with semantic technology.

Fourth, "all participants contribute to the semantic description consistently." Efficient searching for high quality results is based on pertinent matching between well-defined resources and user queries, where the matching reflects user preferences. We use Web site capability description (WSCD) to describe the capabilities of MyPortal and submit user queries consistently.

Fifth, "integrating Web information delivered through Web contents and Web services." Conventional Web contents and Web services have been managed separately as they targeted different consumer, we will support the integrated management of semantic Web contents and Web services at different levels in MyPortal.

2 MyPortal

MyPortal is a "one stop" that links the user to all the information s/he needs. It is at the user's own desktop, which is also a Web server itself and is designed to manage user's personal information with semantic Web technology in a flexible personalized way. It provides both semantic browser and semantic search engine functionalities and these functions manage not only local user desktop information but also the remote semantic MyPortal information. Its information can be published through Web contents and Web services and shared by others with proper authority.

The structure of MyPortal is shown in Fig 1. It consists of following four components: core component provides basic support for semantic Web technologies and knowledge management, user interface component provides a unified interface for creating, browsing, querying, and managing of the relevant information, desktop information management component manages the conventional personal information such as documents, e-mail, contact information, and communication component which is the delegate of the user for communication with other MyPortals.

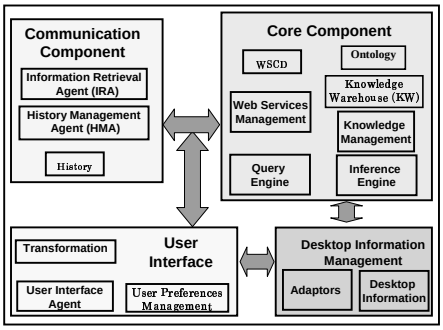


Fig. 1. Structure of MyPortal

One can refer to [4] for a little more detail for MyPortal.

3 Conceptual Architecture of Web Information Retrieval System Based on MyPortal

Our conceptual architecture for a community semantic Web information retrieval system is illustrated in Fig 2.

The architecture consists of three main components: a "consumer" which searches for Web resources, a "provider" which holds certain resources, and a mediator which enables the communication between the consumer and the provider. In our architecture, the providers and consumers are all MyPortal. Each provider describes its capabilities in what we call a WSCD (Web site capability description), and each consumer will submit relevant queries based on user requirements when a Web search is necessary. The mediator is comprised of agents assigned to the consumer and providers using an Agent-Community-based P2P information retrieval method to fulfill the search and access tasks.

3.1 Connecting MyPortals with ACP2P method

The communication between consumer and providers is based on an Agent-Community-based Peer-to-Peer information retrieval method called ACP2P method[2],

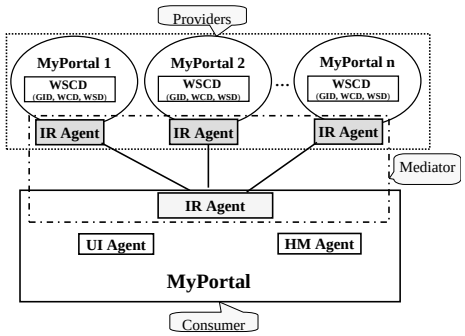


Fig. 2. A Conceptual Architecture

which uses agent communities to manage and look up information related to a user query.

In order to retrieve information relevant to a user query, an agent uses two histories: a query/retrieved document history (Q/RDH for short) and a query/sender agent history (Q/SAH for short). Making use of the Q/SAH is expected to have a collaborative filtering effect, which gradually creates virtual agent communities, where agents with the same interests stay together.

The ACP2P method employs three types of agents: user interface (UI) agent, information retrieval (IR) agent and history management (HM) agent. A set of three agents (UI agent, IR agent, HM agent) is assigned to each user. Although a UI agent and an HM agent communicate only with the IR agent of their user, an IR agent communicates with other users' IR agents to search for information relevant to its user's query. A pair of Q/RDH and Q/SAH histories and retrieved content files are managed by the HM agent.

The ACP2P method is implemented with Multi-Agent Kodama (Kyushu university Open & Distributed Autonomous Multi-Agent) [6]. Kodama comprises hierarchical structured agent communities based on a portal-agent model. A portal agent is the representative of all member agents in a community and allows the community to be treated as one normal agent outside the community.

We are currently planning to use SPARQL RDF query language and SPARQL protocol as our semantic communication interfaces between providers and consumers.

3.2 Web site capability description (WSCD)

Resource location is based on matching between user requirements and Web site capabilities, hence a capability description of MyPortal is necessary. We describe the layered capabilities of MyPortal by layers.

First, we semantically describe the general capabilities of the Web site, and we call this a "general information description (GID)." The GID gives an explicit overview of the Web site capabilities such as their category, topic, and can be used as the initial filter for judging congruence with user preferences. Second, we

give the Web content capability description (WCD), it is the metadata of Web contents and is composed of knowledge bases of all domains involved. Third, we give the Web service capability description (WSD) which is further expressed by two layers: "a semantic Web service description (SWSD)" and "a concrete Web service description (CWSD)." This hierarchical capability-describing mechanism enables semantic and non-semantic Web service capability-describing and matchmaking for different levels.

For the details of our Web site capability description mechanism, one can refer to document [5].

4 Conclusion

In this paper, we addressed our main ideas on constructing a P2P community semantic Web information retrieval system based on MyPortal, mainly focused on how to connect MyPortals to enable automatic and efficient information sharing and what functionalities are necessary when constructing a MyPortal. In the future, we will realize a prototype of MyPortal and a P2P community Web information retrieval system based on MyPortal, and evaluate the effectiveness of our approaches. Experiments in using the ACP2P method for semantic Web data retrieval in a dynamic multiple community environment will also be carried out.

References

1. D. Huynh, D. Karger, and D. Quan. Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF. In *Proceedings of the International Workshop on the Semantic Web (at WWW2002)*, 2002. http://semanticweb2002.aifb.uni-karlsruhe.de/proceedings/Research/huynh.pdf.
2. T. Mine, D. Matsuno, A. Kogo, and M. Amamiya. Design and implementation of agent community based peer-to-peer information retrieval method. In *Proc. of Eighth Int. Workshop CIA-2004 on Cooperative Information Agents (CIA 2004), LNAI 3191*, pages 31–46, 9 2004.
3. L. Sauermann. The Gnowsis Semantic Desktop for Information Integration. In *IOA Workshop of the VM2005 Conference*, 2005.
4. H. Yu, T. Mine, and M. Amamiya. Towards a Semantic MyPortal. In *The 3rd International Semantic Web Conference (ISWC 2004) Poster Abstracts*, pages 95–96, 2004.
5. H. Yu, T. Mine, and M. Amamiya. Towards Automatic Discovery of Web Portals -Semantic Description of Web Portal Capabilities-. In *Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004, LNCS 3387/2005*, pages 124–136, 2005.
6. G. Zhong, S. Amamiya, K. Takahashi, T. Mine, and M. Amamiya. The Design and Implementation of KODAMA System. *IEICE Transactions on Information and Systems*, E85-D(4):637–646, April, 2002.

Authoring and annotation of desktop files in *seMouse*

Oscar Díaz, Jon Iturrioz, Sergio F. Anzuola
{oscar.diaz, jon.iturrioz, jibfeans}@ehu.es

The Onekin Group - University of the Basque Country
P. O. Box 649, P^{o}. Manuel de Lardizabal, 1, 20.018 San Sebastián (Spain)

Abstract. Coping with an increasing number of files is one of the challenges of current desktops. Adding semantic capabilities is one possible solution. Aligned with this proposal, this work introduces the notion of "knowledge folder" as a coarse set of documents bound together by a common ontology. The ontology plays the role of a clipboard which can be transparently accessed by the file editors to either export (i.e. annotation) or import (i.e. authoring) metadata within the knowledge folder. Traditional desktop operations are now re-interpreted and framed by this ontology: copy&paste becomes annotation&authoring, and folder digging becomes property traversal. However, a desktop setting requires seamless tooling for these ideas to get through. To this end, this work proposes the use of the mouse as the "semantic device". Through the mouse, the user can classify, annotate, author, and locate a file as a resource of the underlying ontology. Moreover, being editor-independent, the mouse accounts for portability and maintainability to face the myriad of formats and editors which characterizes current desktops. The "semantic mouse" is implemented as a plug-in for *Windows*.

1 Introduction

Current desktops should be enhanced with mechanisms that permit users to abstract away from les.This work builds on the notion of **knowledge folder**, i.e a coarse set of information elements bound together by a common ontology. The folder contains an ontology, the instantiations, and the resources being annotated. A desktop can hold distinct knowledge folders, and a given le can belong to several knowledge folders. In contrast with current folders, this mechanism attempts to abstract away from how les are physically organised, by providing an ontology-based organisation.

As an example of a knowledge folder, consider all the documentation that goes with a research project. This includes the project proposal (e.g. a Word le), bills being payed by the project funds (e.g. Excel les), etc. These les can be scattered around distinct (physical) folders. Even though, they can belong to the same knowledge folder as some of the following clues indicate,

- data replication among documents (e.g. the funding body appears in the proposal but it is also acknowledged on the articles),
- simultaneous access. More than a document is accessed during a typing session (e.g. when writing the article, the proposal is checked out for the submission deadline),

– event correlation. Creation/removal of the documents are not totally independent (e.g. a bill does not exist without a project proposal), etc.

Current desktops ignore this situation and treat les as isolated units. Today, we copy and paste the text values from one le to another, and the ontology is kept (and managed) in the users mind. And too often, le location turns into digging through a hierarchical folder tree.

This paper presents how this situation can be improved by the introduction of knowledge folders. Speci cally, copying&pasting becomes annotating&authoring, and folder digging becomes ontology traversal. By tapping current le structures into an ontology, authors can both populate the ontology (i.e. annotation), and reuse the instances of the ontology while authoring a document. The ontology instances play the role of a clipboard which can be transparently accessed by the le editors to either export (i.e. annotation) or import (i.e. authoring) metadata within the knowledge folder. As for le location, les are now resources of an ontology. This permits to enhance and contextualize desktop search based on the ontology properties, and navigate along the ontology associations.

Being in a desktop setting, we can not ignore usability. Handling of knowledge folders should be as seamless as possible. Rather than providing separate tools for exporting/importing (i.e. annotation/authoring), we strive to accommodate to the current tools for traditional copy&paste operations: the mouse. This will certainly facilitate user adoption.

To this end, the *semantic mouse* (*seMouse*) is introduced. By clicking on its middle button, *seMouse* exports/imports properties from the *ontology,* regardless of the editor you are working with. It does not matter whether you are working with *Word, PowerPoint, Netscape*, etc, the semantic button is available for annotation/authoring. In this way, the user does not have to move to a new editor when annotating (like in *SMORE* [2]), nor has to learn a new ontological interface when les from different formats are edited (like in *SemanticWord [3]*).

Both, the support of knowledge folders as the underlying infrastructure, and the use of the mouse as the device to interact with this infrastructure, are the main contributions of this work towards making desktops semantic.

Next section introduces *seMouse* through ve scenarios, namely, le classi cation, annotation, authoring, semantic navigation and ontology editing.

2 *seMouse* at work

seMouse is an annotation/authoring device that achieves editor-independence by working at the operating-system level: the mouse. This section introduces *seMouse* with the help of an example.

As a knowledge folder, consider the cluster of heterogeneous documents that goes with a research project. This includes the project proposal (e.g. one Word le), bills payed with the project funding (e.g. twenty *Excel* les), papers as deliverables of the project (e.g. twenty les in both *.pdf* and *.doc* formats), participants (whose desktop counterpart can be either the homepage, an *.html* resource, or a *.pdf* resource) and comments (being realized as either emails or .doc resources).

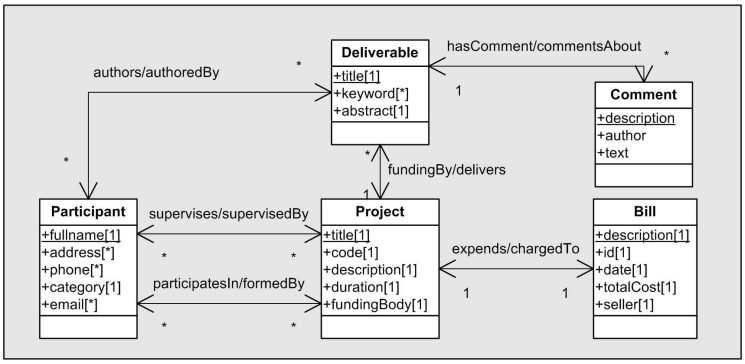


Fig. 1. A sample ontology.

Regardless of their format and folder location, it is likely that a high degree of content reuse as well as frequent contextual navigations within this le space happens. *This is what makes this set of files a knowledge unit.* Being in a participant -an *html* le-, you frequently need to locate her project proposals -Word les-, or being in a project proposal, the associated papers -*PDF* les- are commonly accessed.

A knowledge folder comprises an ontology (gure 1 shows the one for the sample problem). Five classes are identi ed. Each class is characterized by a set of value-based properties (e.g. *title, keyword, abstract*). Associations are de ned between these classes (e.g. a project is *supervisedBy* a participant) (termed *ObjectProperty* in OWL). And the expressiveness of OWL can be used to de ne inverse and transitivity properties between the associations.

Although the ontology is at the core of the semantic desktop, this paper focuses on authoring and annotating resources of the ontology. We do not address how the inference power of the ontology can achieve its full potential in a desktop setting.

Once the ontology has been set, the population process begins. The key idea is to use the mouse as the semantic device so that interactions with the underlying ontology are achieved via mouse clicks. Speci cally, pressing the middle button on the mouse causes an interaction with the ontology manager. This interaction is context-sensitive, i.e. the button accomplishes distinct operations depending on the place the pointer sits on. Next paragraphs introduce ve scenarios of the use of the semantic mouse.

Scenario 1: file classification. First of all, les need to be identi ed as instances of any of the ontology's classes. This is achieved by opening a le, and pressing the middle button. A menu pops up for the user to indicate to which class this le is a resource.

Scenario 2: annotation (see gure 2 and 3). Annotation&authoring becomes the counterpart of copy&paste in traditional desktops, with the difference that now these operations are conducted along the ontology net. What is being exported(i.e. copy) is no longer a string but a class property of the ontology.

If a le has already been categorised, the annotation process may begin. If some text is selected, the mouse is used to export this text as part of the value of a property as it is shown in gure 2. Of course, the set of properties will depend on the class of

Fig. 2. Scenario 2: annotation. Some text is selected. Being a *deliverable* file, the menu displays properties of this class. The text will become the value of the chosen property.

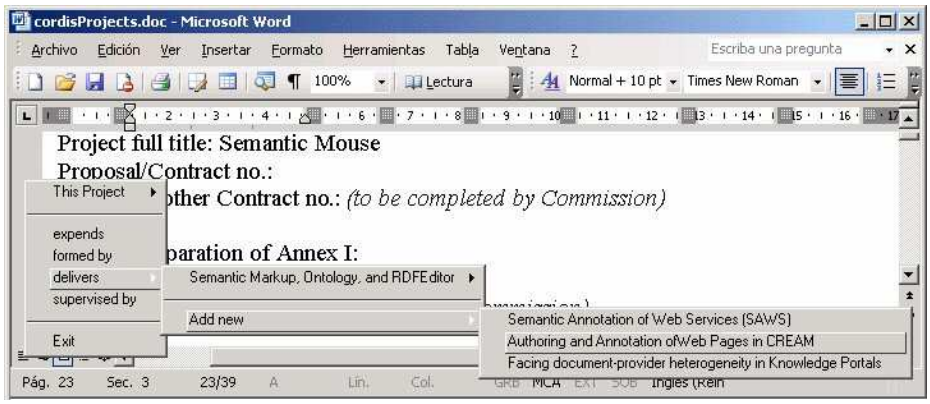


Fig. 3. Scenario 2: annotation. No text is selected. The menu shows associations of the file class.

the resource. In the example, *title, keyword* and *abstract* correspond to properties of the *deliverable* class.

On the other hand, if no text is selected, the middle button is used to establish associations with other les. This situation is exempli ed in gure 3. In this case, the CORDIS project template for EEC projects has been used. When the middle button is pressed, a menu pops up for the user to link the current resource with other target resources. The menu is customised for the current resource, that is, the associations are restricted to those available for the current resource, whereas the target les are also limited to those of the appropriated class. In the example, the association can only be established with *deliverables* les since this is the destination class of the *delivers* association.

Scenario 3: authoring (see gure 4). Associations being set during annotation can now be exploited. For instance, the *project* resource can import the title of its associated *deliverable* resources. In the example, the article *"Authoring and Annotation of Web Pages in CREAM"* appears as a *deliverable* of the current le. By selecting this article,

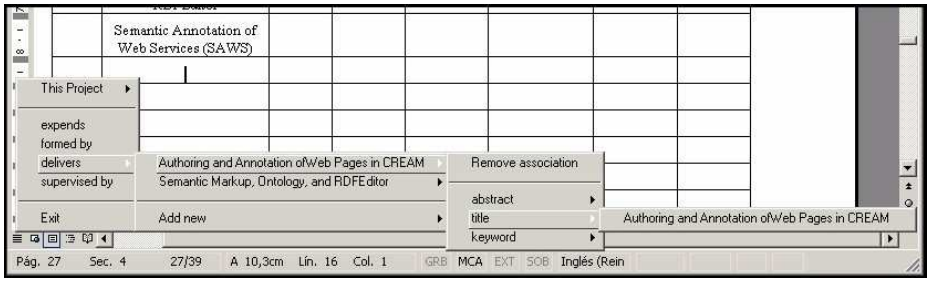


Fig. 4. Scenario 3: authoring. The *title* of a *deliverable* is imported into a *project* resource.

the menu is extended rightwise to show up its properties. The user can select one of these properties, and its value is inserted at the cursor place.

Scenario 4: semantic navigation . File location in current desktops frequently implies folder digging. By contrast, semantic navigation strives to exploit the associative behaviour of the human memory. A resource can be located from the resources it is related to. That is, the ontology provides the context to facilitate resource location.

Once a le has been selected, semantically-related les can be located by pressing the middle button, regardless of the folders where these les are physically located, providing a resource-centric navigation. This facilitates location of neighbour resources, but it may be cumbersome whenever browsing is required. In this case, a graph-based RDF visualizer can be a better option (see [1] for an overview of RDF visualizers).

3 Conclusions

This work strives to lower the adoption barrier of the semantic desktop by providing seamless tooling. To this end, we support the notion of knowledge folder as the underlying infrastructure, and the semantic mouse as the interactive device. Being editor-independent, the mouse accounts for portability and maintainability to face the myriad of formats and editors which characterizes current desktops. Similar to other areas of computing, a balance is needed between generality (e.g. format-independence, editor-independence, etc), and functionality (i.e. the semantic tooling available). *seMouse* illustrates a semantic-lite approach where a compact set of functions are available to no matter which editor within *Windows*.

References

1. John Gilbert and Mark H. Butler. Review of existing tools for working with schemas, metadata, and thesauri. Technical report, Hewlett Packard Laboratories, October 2003.
2. Aditya Kalyanpur, James Hendler, Bijan Parsia, and Jennifer Golbeck. SMORE - Semantic Markup, Ontology, and RDF Editor. http://www.mindswap.org/papers/SMORE.pdf, 2004.
3. Marcelo Tallis. Semantic Word Processing for Content Authors. In *Workshop Notes of Knowledge Markup and Semantic Annotation Workshop (SEMANNOT 2003). Second International Conference on Knowledge Capture (K-CAP 2003)*, October 2003.

Context as Foundation for a Semantic Desktop

Tom Heath, Enrico Motta, Martin Dzbor

Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes, MK7 6AA, United Kingdom
{t.heath, e.motta, m.dzbor}@open.ac.uk

Abstract. Adoption of semantic web technologies and principles presents an opportunity to change the conceptual model of desktop computing. Moving from a traditional position where the desktop is largely tied to a specific computational device, a semantic desktop could exist as a broad, networked space defined relative to the user. In this position paper we argue that personal, computing, and knowledge contexts are the appropriate means by which to define and shape the desktop space, and that collectively they provide the foundation for novel functionality in a semantic desktop.

1 Introduction

The traditional conceptual model of desktop computing is no longer applicable. In a heavily networked world, the distinction between a desktop as realised by an operating system running on local hardware, and the web as something separate and external, is false. What makes the desktop unique is not the location of execution of its underlying code, but its role as a space that is largely under the control of one individual; a personal domain that in some way reflects or represents their world.

Consequently, developers of a semantic desktop face a choice between simply applying semantic web principles to a traditional desktop environment whilst retaining the same conceptual model, or using the power of these technologies to enable a more ecologically valid interaction paradigm that emphasizes the person, the tasks they perform, and the context in which they do so, irrespective of where the code is executed. Whilst both approaches may bring benefits, we advocate the second. In this position paper we will discuss how that context may be defined, represented, and used in pursuit of this aim.

Ubiquitous computing literature has largely defined context in terms of computing resources, the user's location, and the identity of people around them [1]. Our conceptualization of a semantic desktop requires a broader view to be taken if this greater ecological validity is to be achieved. In the following sections we will identify three facets of context (personal, computing, and knowledge) and the interactions between them, whilst also examining how each may be represented and utilised in a semantic desktop, and what novel features this may provide.

2 Personal context

Reminiscent of a phenomenological view of context [2], personal context encompasses properties of the individual as they exist in the world, which may influence a task being performed but are not specific or unique to it. In [3] the authors identify several such properties, including a person's *social context* (social networks they are part of), their *preferences* (values or opinions held), *resources* they have available for performing a task (such as time or attention), and their *location*. Inevitably these factors interact, such as one's location in a busy office reducing the attention available for performing a task.

In terms of semantic representation, social networks and nodes within them can be represented with vocabularies such as FOAF [4] and SWAP Personal Information Markup [5]. Representing the resources available for performing a task may be informed by work in the field of instant messaging, regarding how to describe presence and availability (e.g. online, low attention, do not disturb). Personal preferences might be described using multiple domain-specific vocabularies (such as the Vegetarian Ontology [6]) or through a generic ontology of preference which could be applied to any domain. Vocabularies such as Basic Geo [7] can be used to express latitude and longitude locations in a machine-readable way. However, to meaningfully represent the individual's location this will need to be complemented by ontologies of place and space that describe locations in terms of their function, significance, and interrelations.

At present these personal context factors, if represented or used at all within a desktop environment, are done so on a per-application basis, adding unnecessary cognitive overhead to the performance of tasks. For example, a user may have to maintain separate contacts lists in their email and instant messaging applications, or may have to state their location when searching the web for local restaurants and again when booking a flight.

The ability to create personal context widgets to represent and manage this context information and make it available across tasks is a key benefit of a semantic desktop over a conventional one. We hypothesise that this provides a more realistic cognitive model to the user, where the factors and entities that characterise their world are captured in one place rather than distributed piecemeal or not represented at all. A trend towards integrating applications that share a social component can be seen in Chandler [8], and this should be welcomed as long as the focus remains on integrating contextual information and providing services on top of this.

3 Computing context

In the ubiquitous computing literature, resources such as network connectivity [9], applications available, and characteristics of the device being used are seen as key aspects of context. These factors should be represented and utilised in a semantic desktop as they may impact on how a task is performed, and they do reflect aspects of the user's world. This may be achieved using CC/PP profiles [10], which provide a means to describe the capability of a device and preferences about how it is used, and

may be extensible to describe factors such as the network connectivity available at any one time.

An ontology of application types that describes their capabilities in functional terms (e.g. *ImageManipulator*, *MessageHandler*) could be populated with those applications available to any particular computing environment (whether they are implemented at the local machine or the *webtop* level).

However, our view emphasises the desktop as a reflection of the user's world. Consequently, the objects that populate that world and the tasks performed within it are key, with the resources and applications available playing only a secondary, more abstract role. Influenced by aspects of the Xerox 'Star' computing environment [11] we believe that a semantic desktop can enable an interaction paradigm centred on digital objects, their contexts, and associated tasks, in a way that a conventional desktop cannot. By object context we mean factors such as the people or objects depicted in a photograph or the occasion on which it was taken, the person who sent a message, or the organisation who published a document.

Whilst some actions users may wish to perform would be generic to most objects (such as *view*, *share*, *edit*), others may be determined by the object type (such as cropping or resizing an image), or its associated contexts. Awareness of how context manifests itself for different types of object would allow true *context menus* to be implemented, providing access to functionality or services tailored to that specific object. For example, such a menu for an audio file could provide access to other tracks by the same artist, a discography, or current tour dates. Similarly, a context menu for a *message object* might provide access to contact details for the sender, or inferences about how best to contact them in response, based on their current availability.

Ontological descriptions of the types of objects available in a semantic desktop, the actions that may be performed on them, their associated contexts, and the capabilities of applications available to the desktop could enable a semantic registry for the desktop, with reasoning able to determine the best code to execute to perform a particular action or task, rather than requiring an explicit decision from the user.

4 Knowledge context

Whilst a traditional filesystem is concerned with managing files that exist on a local machine, the same limitations do not apply to a semantic desktop. Instead, a semantic filesystem should focus on the management and application of knowledge that supports user tasks wherever it resides, enabling a knowledge-oriented computing environment that adheres to our model of the desktop as a representation one person's world.

Here semantic web technologies can bring great benefits over conventional desktops, firstly through improved knowledge representation, secondly through the ability to reference any resource, irrespective of whether it is a digital object itself or simply a reference to an entity in the real world, and thirdly irrespective of whether it is located locally or remotely.

However, if these abilities are to be maximally exploited in the support of tasks, then we must be able to assess the provenance and validity of knowledge in the system; the knowledge context. Just as conventional filesystems have metadata recording when a file was created and last modified etc., so a semantic filesystem should have knowledge metadata indicating the source of knowledge in the system, its age, whether it has been validated or not, and inferences about its likely trustworthiness. An ontology of knowledge characteristics could define the exact nature of this metadata, populated over time as knowledge is added to the system and evidence is accumulated to qualify, validate or contradict the assertions.

Awareness of knowledge context brings some novel features to a semantic desktop which could support the tasks outlined in [3]. If *locating* a certain piece of knowledge within a semantic desktop, the ability to prioritise results based on inferred trustworthiness could help reduce cognitive overhead. Similarly the user may only wish to *monitor* knowledge within the environment that is recent whilst ignoring older stable items, and knowledge context can enable this. It also provides a foundation for the *evaluating* task, which consists of "determining whether a particular piece of information is true, or assessing a number of alternative options".

5 Interactions between facets of context

Whilst the context facets discussed here have been treated separately there are inevitable interactions between them, and a semantic desktop must facilitate these to provide maximum benefit. For example, how well someone is known to us (personal context, social factor) is likely to effect how much we trust knowledge they share with us (knowledge context). Similarly, in a situation where someone has limited resources for performing a task they may accept knowledge as a solution even if it is from a less trusted source, simply to have reached some solution. These interactions can enable novel features, as shown by the example above where an objects context menu could provide access to the author's contact details and, where permission exists, to their personal context, such as current availability and inferences about how best to contact them.

The extent of these interactions can be illustrated by the task of locating recipes for a dinner party. In this case a semantic desktop could take into account the personal context of the user by excluding certain cuisines they have indicated they don't like, and by prioritising results from the same sources as recipes that have a trusted knowledge context. An object context could be provided by creating links between the semantically annotated recipes and the necessary ingredients as listed in an online shop.

6 Conclusions

A semantic desktop has the potential to introduce a new style of interaction in personal computing that is not feasible with conventional technologies. In this knowledge-based environment, the desktop is defined in relation to the user, not the

hardware, operating system, application, or protocol being used. However, parameters are required to shape this semantic desktop to the individual, and in this paper we have argued that context is an appropriate and powerful basis on which to do so. Whilst each facet of context we have discussed enables different functionality, interactions between them are central to the performance of tasks by the user. A semantic desktop based on the foundation of these facets and their interactions represents a novel and powerful interaction paradigm.

Acknowledgements

This research was partially supported by the Advanced Knowledge Technologies (AKT) project. AKT is an Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

The authors would also like to thank John Domingue and Marian Petre, conversations with whom inspired some of the ideas in this paper; and two anonymous reviewers, whose comments led to significant improvements.

References

1. Dey, A. K., Abowd, G. D., Salber, D.: A Conceptual Framework and a Toolkit for Supporting Prototyping of Context-Aware Applications. Human-Computer Interaction 16 (2001) 97-166
2. Svanaes, D.: Context-Aware Technology: A Phenomenological Perspective. Human-Computer Interaction 16 (2001) 379-400
3. Heath, T., Dzbor, M., Motta, E.: Supporting User Tasks and Context: Challenges for Semantic Web Research. In: Proc. ESWC2005 Workshop on End-User Aspects of the Semantic Web (UserSWeb) (2005)
4. Brickley, D., Miller, L.: FOAF Vocabulary Specification. http://xmlns.com/foaf/0.1/ (2005)
5. SWAP Personal Information Markup. http://www.w3.org/2000/10/swap/pim/doc (2003)
6. Golbeck, J., Parsia, B., Hendler, J.: Trust Networks on the Semantic Web. In: Proc. Cooperative Intelligent Agents (2003)
7. Basic Geo (WGS84 lat/long) Vocabulary. http://www.w3.org/2003/01/geo/ (2003)
8. O. S. A. F.: What's Compelling About Chandler: A Current Perspective. http://www.osafoundation.org/Chandler_Compelling_Vision.htm (2005)
9. Khedr, M., Karmouch, A.: Negotiating Context Information in Context-Aware Systems. IEEE Data Engineering Bulletin (2004) 21-29
10. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/ (2004)
11. Johnson, J., Roberts, T. L., Verplank, W., Smith, D. C., Irby, C., Beard, M., Mackey, K.: The Xerox Star: A Retrospective. Computer 22 (1989) 11-26, 28-29

DeepaMehta – A Semantic Desktop

Jörg Richter[2] and Max Völkel[1] and Heiko Haller[1]

[1] AIFB, University of Karlsruhe, Germany
{mvo,hha}@aifb.uni-karlsruhe.de,
http://www.aifb.uni-karlsruhe.de/WBS
[2] Co-Founder and Lead Architect of DeepaMehta, Berlin, Germany
jri@freenet.de, http://www.deepamehta.de

Abstract. DeepaMehta is an open source semantic desktop application based on the Topic Maps standard. Its conceptualization and especially the UI have been guided by findings of cognitive psychology in order to provide a cognitively adequate working environment for knowledge workers of all kind. It uses a graph visualization similar to concept maps. DeepaMehta aims to evolve nowadays' separated desktop applications into an integrated workspace, enabling the user to organize, describe, relate, edit and use almost any information objects.

Introduction In this paper we present the Topic-Map-centric semantic desktop „DeepaMehta". First we state some psychological requirements for personal knowledge management (PKM). Then we describe the UI concepts and their realisation via the Topic Map metaphor. We conclude with a brief evaluation based on psychological criteria and a selection of related works.

Psychological Requirements It should be the main goal of any knowledge management software, to *facilitate the creation, externalisation, and (re)construction of knowledge*. Since there is evidence, that conceptual human knowledge is actually stored in an *associative* way, comparable to semantic networks [1], it appears sensible to provide the knowledge worker with a UI, where the contents are displayed, managed, created, and refined in such an associative manner (i. e. items together with their relations to other items) that enables the construction of semantic networks [2].

Mapping Techniques[3] exploit the fact, that one can use his natural sense of spatial orientation, which has not been optimized by evolution for text or even hypertext. We can use our sense of orientation that easily distinguishes spatial positions and layouts (also in a plane) to gain orientation in our *knowledge space*. Furthermore, there has been a lot of research in cognitive psychology that showed, that the use of concept-map-like techniques can have various positive effects on learning and problem solving – i. e. knowledge generation and -use [2–4].

[3] In this article the Term "Mapping" is used as coined in the domain of instructional psychology, i. e. in the sense of *creating and using visual knowledge representations* called "maps" like mind-maps, concept maps etc.

Another requirement stemming from psychological viewpoint is a low *cognitive overhead*. This is "the additional effort and concentration necessary to maintain several tasks or trails at one time" [5]. Because our working memory and thus capacity for conscious processing are so limited, we should avoid wasting it to secondary tasks like worrying about saving files, dealing with layout and formatting while writing or regaining orientation in the information environment. It is thus a major goal of usability research to reduce such cognitive overhead.

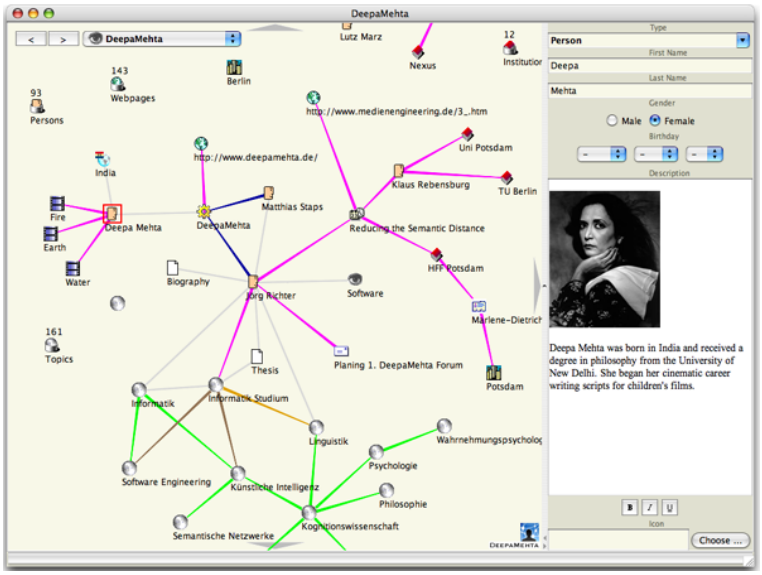


Fig. 1. A Topic Map (typical DeepaMehta working screen)

Design The design of DeepaMehta is centered around the model of *Topic Maps*. Topic maps are a human-oriented approach to encode knowledge and knowledge about knowledge. Topic maps consists of *topics*, *associations* and *occurrences*. In the semantic web, this relates to *resources*, *relations* and *instances*. Topic maps form a type system which is self-describing, much like RDF schema. We assume some familiarity with Topic Maps and refer the reader to the ISO standard [6] and relations to RDF [7].

DeepaMehta is an application framework with a Topic-Map-based user interface (c. fig. 1), the design of which is based on findings in cognitive psychology. Information of any kind as well as the relations between information items can be displayed and edited in the same space. The user is no longer confronted with files and programs. There are no overlapping windows, no menu bar and no dialog boxes. Topic Maps are individual views on interconnected contents.

An application in this context is a collection of topic types, for which specialised and generic commands (e. g. what's related, hide, retype, delete) are executable by the user.

A Cognitively Adequate User Interface One of the most obvious problems in current desktop user interfaces is that of context switching. Users are currently switching between different applications for every sub-task. Each switch presents a completely new interface to the user, even if the underlying concepts don't change much: An address used within a text document is *conceptually* not much different from *the same address* used in an email. Today's desktop UI's are application-oriented, not data- or task-centric. DeepaMehta gives the user **stable views**, letting the user focus on the task itself, without leaving the work-context: In one and the same view the user can read an email, link it to an existing topic, attach a note to it, search for related media, save the search results, make semantic statements and spatially arrange all these items on the screen. If a user leaves his *workspace*, he will find it later exactly as he left it.

Browsing the WWW is an easy endeavor. Figuring out later where one has been is not that easy. A browsers history is purely time-based (1) and offers no possibility to attach any kind of information to it (2). Even worse, after a fixed time interval, the history is often erased automatically (3). If not, it becomes so large that a useful information retrieval is impossible (4). Additionally, the browsers history contain no information about other resources accessed (5). Bookmarks do offer the ability to attach notes to web addresses, but still have the problems (4) and (5). Bookmarks require a lot of additional effort compared to normal web browsing. DeepaMehta offers **constructive browsing** as a solution. Each resource used is represented as a topic in the current workspace. Each new topic is placed right next to the one where it originated from. The user can replace it conveniently to another place in his workspace, which is always visible. Now surfing the web – or accessing other resources – automatically creates a map of seen things. Even searches, refined searches and search results are represented in the same intuitive fashion. This spatially arranged map visualises a work process better then a few named bookmarks or a list of URLs. This map is automatically persisted and usable.

Traditional applications have a fixed set of objects they deal with. The DeepaMehta **type system is extensible**. The user can construct new topic and relation types on the fly in the same user interface, using a set of very few built-in topic types like *topic type*, *assoc. type*, *property*, *prop. value*, *data source*, *search*, *map*, and *workspace*. New topic types can be used instantly and serve as the basis for the UI. The user management is also carried out using the topic map concept (e. g. each user represented by a topic, with relations to groups and shared workspaces).

DeepaMehta is realised with a **service oriented architecture** (SOA) offering many communication and integration options. It comes with a rich client, a browser interface and a set of desktop adapters. Data can be stored in several back-ends, exported to XTM or PDF or even shared via SOAP web services.

Each topic type can be provided with it's own java class, to give it unique functionality.

For **collaboration**, users can share their workspace (topics, topic types and types positions) in the web interface with others, relying on the integrated access control system and central synchronisation server.

Evaluation The DeepaMehta architecture defines a **new application model** and gives developers a framework to design DeepaMehta-applications. Such applications are easy to maintain and updated, as the business logic resides on the server. Also a range of interaction front-ends is offered: rich client, thin client, web browser and even a PDA interface[4]. The thin client framework provides a solid framework for many kinds of interaction clients. A more up-to-date user interface is planned.

This flexibility enabled the **successful deployment of several commercial sites** in a variety of domains[5]. Among these are two eLearning projects, a geographic information system about city quarters "Kiezatlas". For consultants, a competence analysis tool was implemented. A third project acts as an information management system for modern and contemporary artwork.

In order to evaluate the usefulness of the innovative **user interface** for personal knowledge management, we check[6] it against a set of criteria set up to evaluate visual mapping tools for personal knowledge management from a cognitive psychological point of view [4].

- *Free Placing* an item on the canvas is possible.
- *Free Relations*: Stating relations between items in DeepaMehta is possible in all degrees of formality (unlinked nodes, unlabeled links, labelled links, typed links)
- Every item can be given an *Annotation* in natural language.
- The most basic and useful way, to deal with complexity and clarify the macro structure of a domain, is to use *chunking a.k.a. clustering* [8, 9]. DeepaMehta offers only *visual* grouping within a single map or the ability to create explicit sub-maps.
- For brainstorming, too many mouse-clicks are currently required and creating new items without leaving the keyboard, is not possible at all. The *cognitive overhead* should be reduced for standard tasks.
- The *detail and context problem* [5, 10] is solved in DeepaMehta by splitting the screen and showing the user always both the actual content (property pane) and the context information (topic pane). The linked topic-view on the left gives him always all related items, while the rights side can be navigated classically by traversing links or editing property values.

[4] Download at `http://www.deepamehta.de/docs/deepamobil.html`

[5] `http://www.kiezatlas.de`, `http://artfacts.net`

[6] Due to space limitations, the commented evaluation can only be found in the long version of this paper

DeepaMehta's UI and interaction paradigm takes a consequent approach of minimalist design, where only relevant controls are shown. As this differs from common interfaces, it requires some initial time to get acquainted. Providing zooming capability and a grouping feature would surely increase its utility, especially for the use of larger and more complex maps.

Conclusion In this paper we presented the DeepaMehta framework for a unified personal knowledge management. A user works with his *unified knowledge* (**UK**) as he sees data bases, contacts, emails, projects and other data through the same structured, interface. This interface offers *personal ontology-based management* (**POM**) through its flexible and extensible Topic Map approach. The whole application is part of a *networked environment* (**NE**), in which users can easily share concepts and content. Thus DeepaMehta is a true **NEPOMUK**, which has been used successfully in a number of commercial projects. In the future, usability improvements and RDF integration will be explored.

Related Work The node-and-link type of DeepaMehta's visualisation is inspired by the *concept capping* approach [11], that has proven successful in improving learning in many different scenarios (see [3] for an overview of studies). There has also been a lot of research on a semi-formalized derivative of concept mapping, dubbed *knowledge mapping*, that uses fixed sets of typed relations [2].

Acknowledgments: Research reported in this paper has been partially financed by the EU in the IST-2003-507482 project Knowledge Web [7] and is supported by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project. We would like to thank our colleagues for fruitful discussions.

References

1. Quilian, M.R. In: Semantic Memory. MIT Press, Cambridge, MA (1968)
2. O'Donnell, A.M., Dansereau, D.F., Hall, R.: Knowledge Maps as Scaffolds for Cognitive Processing. Educational Psychology Review **14** (2002)
3. Jonassen, D.H., Beissner, K., Yacci, M.: Structural Knowledge: Techniques for Representing, Conveying and Acquiring Structural Knowledge. Lawrence Erlbaum Associates, Inc (1993)
4. Haller, H.: Mappingverfahren zur Wissensorganisation (2003)
5. Conklin, J.: Hypertext: an introduction and survey. Computer **20** (1987) 17–41
6. Michel Biezunski, Martin Bryan, S.R.N.: ISO/IEC 13250:2000 Topic Maps. Technical report (1999)
7. Pepper, S.: Ten theses on Topic Maps and RDF. Technical report (2002)
8. Anderson, J.R.: Cognitive Psychology and Its Implications. 6th edn. Worth Publishers (2005)
9. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review **63** (1956) 81–97
10. Furnas, G.W.: Generalized fisheye views. (1986) 16–23
11. Novak, J.D., Gowin, D.B.: Learning how to learn. Cambridge University Press, New York (1984)

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

How to build a Snippet Manager

Steve Cayzer, Paolo Castagna

Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol BS34 8QZ
{steve.cayzer, paolo.castagna}@hp.com

Abstract. In our research group, there is a need to capture, organize and share resources associated with a domain of exploration. We are building a tool for this task, based on previous experience in the knowledge management domain. In this position paper, we present our thoughts on what works (and what doesn't work), together with details of our initial implementation.

1 Introduction

The snippet manager idea is not a new one [1]. It refers to the idea of a small peer group capturing 'snippets' of information in a lightweight manner, categorizing and sharing them. There have been a number of approaches to this problem over the years; in this paper we present our opinions on what works, and what doesn't work. We also define the scope – even the ideal snippet manager would not be a panacea for knowledge management generally. Rather, it is a useful tool (or at least a useful concept) for a specific task. We have started to implement a tool using the principles outlined here, and we present some design details. We also introduce the success factors that we intend to adopt, and that we hope will be more generally useful.

2 The snippet manager problem domain

Knowledge management is defined widely, for example achieving a global sharing of knowledge within a company [2]. However, our domain of interest is rather more tightly defined:

"In [my group] we frequently circulate items of interest (such as news articles, software tools, links to Web sites, and competitor information). We call them ***snippets****, or information nuggets, we would like to store, annotate, and share. Email is not the ideal medium for these tasks; its transient nature means the snippets are effectively lost over time. Yet the risk from using a more formal process, like a centralized database, is that it is both cumbersome to use (a barrier to entry) and overly rigid in its data model (not amenable to storing different types of information). Our need illustrates what I call decentralized, informal knowledge management…"* [3]

The point here is that the domain allows us to make some simplifying assumptions. Firstly, the group is small, often co-located. Members can 'pop round' to discuss ideas or gather for an informal discussion. In the particular group we are designing for, there are regular, weekly meetings. So we assume that information flow is unhindered, and that conflicts or inconsistencies can be quickly ironed out. We need not rely on snippet manager as the sole conduit for communication. A small group also makes the job of converging on a domain model much easier. We do not assume that we will get the model(s) right first time, but the first pass should be good enough to get general buy-in, and improvements can occur by means of incremental evolution.

Secondly, the users are technically literate. This means that they are likely to quickly get to grips with a new tool, and may be motivated to make some small changes to behaviour for sufficient added value (a good example of this would be the use of 'graffiti' writing on PDAs). However, getting the balance right is not always easy; we note that in our (internal) semantic wiki, people hardly ever use the supplied wiki syntax to add RDF metadata. This may be due to the lack of instant, direct reward for adding metadata, a point which we are trying to address in our work.

Finally, the domain of interest is tightly focused, even more so than that envisaged by Cayzer [3]. Our laboratory is interested in a myriad of semantic web related topics, but the user group for the current incarnation of the snippet manager is specifically and actively looking at a particular topic, that of enterprise information management.

2.1 Use cases

What is it, then, that the snippet manager is expected to achieve? We are in the early stages of this project, but we have engaged the user community and gathered some initial use cases in order to inform our guiding principles, a few of which are described here. We reiterate that these principles are relevant for our domain of interest – small group, tightly focused, technically literate researchers. We don't expect the principles to necessarily generalize to the whole of knowledge management, or to the web at large. However, we believe that our problem domain is sufficiently common for these principles to be of value for the semantic web community.

Easy Capture
"I need a way to collect evidence (web pages, PDFs, emails, forum posts), and to categorise parts of these so they can be linked together for post-hoc search."
"I need a way of annotating resources with evidence - 'why have you written this'"
It should be ludicrously simple to collect snippets, using familiar methods such as bookmarklets or email. Snippet Manager should also pull in snippets from other sources (eg intranet databases) and handle provenance.

Editable Ontologies

I need to create a category or classification for a new area of interest. Now, I need to add new companies, products, documents or links and tag them with this classification. I want to create relationships between these instances - e.g. competitor links.
Our users will certainly want to change the ontologies on the fly. Although this sounds like a tall order, in our case both the structure of the ontologies (effectively taxonomies) and the nature of the changes (adding/removing/renaming a node) can simplify the implementation enormously. Of course the UI for such changes may not be trivial.

Export
I want a regular alert showing the results of a web search for a topic. [OR I want to produce a report that shows all relevant products or technologies for a given topic]
From a technical point of view, the ability to export (meta)data in a standard, machine readable way is a future-proofing mechanism, intended to prevent the portal becoming yet another information silo. From a user point of view, export in a *human-readable* form is equally important.

Web Application
Our experience with the early snippet manager prototype [1] taught us that there is considerable reluctance to download software, let alone to standardize on it across a group. In addition, the snippet manager should be integrated into a users' normal work pattern. For our group, this suggests a web application such as a portal.

Immediate Feedback
There should be an instant reward for the user who adds metadata. The community aspect should be (from the user's point of view) a beneficial side effect.

4 Implementation Details

We have used the semantic portal [4] idea and codebase to provide a browsing interface over the group's snippets. Essentially, this portal uses the metadata to drive a facet browser, so that users can find what they are looking for using a variety of search paths. We are building simple capture modalities such as bookmarklets, mail processors and web forms; and importers for other systems such as blogs, technical reports, people databases and the group's official wiki. For export, we plan RSS feeds, email alerts, customizable reports and a SPARQL[5] interface for programmatic access.

4.1 Success Factors

We have previously built several semantic web applications whose primary function was to demonstrate a particular aspect of the technology. Our focus here is to build a tool. Therefore the simplest way of assessing its success is to measure its usage:

1. At what rate is new content added to the snippet manager?
2. What proportion of the user group use the snippet manager as a day to day tool
3. How well does the snippet manager integrate with other tools in use?
4. How often is the snippet manager consulted for information or report generation?
5. What is the satisfaction level of the users?
6. How quickly can new user requirements be integrated into the tool

These measures are largely qualitative in nature. Yet they get to the heart of what of means to build a semantic web tool for personal, and group, productivity. We intend to assess our work using these criteria.

5 Related Work

Simile's Semantic Bank [6] is a snippet repository that lets you persist, share and publish data collected by individuals, groups or communities. Data capture is accomplished using a Firefox extension called Piggy Bank [7], and the information is accessed using a faceted browser. It is probably the closest system in philosophy to ours, but there are some important differences. Firstly, Semantic Bank is intended to be a general purpose, potentially global scale snippet repository. This means that there are significant research challenges in making the ontologies both sufficiently compact and understandable. In snippet manager we chose to have a small number of tightly focused facets. Secondly, our aim is to allow both the gathering of snippets and the linking of these snippets with data from other sources.

The broader idea of a semantically enabled website is explored in a number of public portals, notably the Semantic Web Community Portal [8] and MindSwap [9], both of which use metadata for filtering and querying. As the number of items increases, the value of our faceted browsing approach becomes more apparent. There are other public portals such as Ontaria [10] and SchemaWeb [11], which are primarily intended for browsing ontology data.

Many people use their weblog as a knowledge management tool and we think that structuring the content of a post by adding some metadata could be useful for a group of people. But the chronological view that weblog gives to the content not always it the best solution to let users move through information. Our solution to this, which we call semantic blogging [3], uses metadata guided views, such as record cards or tables. A similar approach has been taken by the structured blogging community [12]. A more subtle point is the information model, in which the blog entry is no longer the primary object. Rather, the information item (such as web page, report or person) which is being blogged about takes centre stage. The blog entry is an annotation attached to this item. Armed with this perspective, the semantic blog becomes a useful personal knowledge management tool, and a source of data for the snippet manager.

Wikis are also interesting tools for collaboratively building knowledge, and there are examples [13, 14] that use metadata to enhance navigation and to provide multiple views. In some ways the snippet manager idea is similar (although our data entry mechanism is different); however we integrate information from a number of sources. Just like blogs, wikis are a valuable source of data for the snippet manager.

6 Conclusion

In this position paper, we have outlined our thoughts on what it would take to build a snippet manager for small group domain-focused knowledge sharing. We have shared some design principles which we hope will prove generally useful. We have also explained how we are going about building a system using these ideas. We have high hopes that our user-centred approach will function less as an interesting demo and more as a genuinely useful tool.

References

1. Banks, D., Cayzer, S., Dickinson, I., Reynolds, D. The ePerson Snippet Manager: a Semantic Web Application. Hewlett-Packard Laboratories Technical Report HPL-2002-328 (2002): http://www.hpl.hp.com/techreports/2002/HPL-2002-328.html
2. Davenport, T. H. and Prusak, L. Working Knowledge: How Organizations Manage What They Know. (1997) Harvard Business School Press.
3. Cayzer, S. Semantic blogging and decentralized knowledge management. Communications of the ACM 47, 12 (Dec. 2004), 47-52.
   DOI= http://doi.acm.org/10.1145/1035134.1035164
4. Reynolds, D., Shabajee, P., Cayzer, S., Steer, D. Semantic Portals Demonstrator - Lessons Learnt. Sept 2004. http://www.w3.org/2001/sw/Europe/reports/demo_2_report/
5. Prud'hommeaux, E., Seaborne, A. SPARQL Query Language for RDF. W3C Working Draft 21 July 2005 http://www.w3.org/TR/rdf-sparql-query/
6. The Semantic Bank project http://simile.mit.edu/semantic-bank/
7. Huynh, D., Mazzocchi, S., Karger. D. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. Submitted to 4th International Semantic Web Conference (ISWC 2005)
8. The Semantic Web Community Portal http://beta.semanticweb.org/
9. The MindSwap Group http://www.mindswap.org/
10. The Ontaria project http://www.w3.org/2004/ontaria/
11. SchemaWeb: RDF Schemas directory http://www.schemaweb.info/
12. Structured blogging http://structuredblogging.org/
13. Aumueller, D.Semantic authoring and retrieval within a Wiki. Demonstration track, 2nd European Semantic Web Conference (ESWC 2005). http://wiki.navigable.info
14. Tazzoli, R., Castagna, P., Campanini, S. E. Towards a Semantic Wiki Wiki Web. 3rd International Semantic Web Conference, Poster Track (ISWC 2004).
    http://platypuswiki.sourceforge.net/

HyperSD: a Semantic Desktop as a Semantic Web Application

Daniel Schwabe, Daniela Brauner, Demetrius A. Nunes, Guilherme Mamede

Departamento de Informática, PUC-Rio, Brazil,
{ dschwabe, dani, mamede}@inf.puc-rio.br
demetrius@interface-ti.com.br

Abstract. In this paper, we show how one can leverage a Semantic Web application development environment to define an application that is a Semantic Desktop browser called HyperSD. In addition, special wrappers have been defined to allow importing regular desktop objects, such as files, person records, calendar entries, etc… into the semantic desktop accessed through HyperSD.

Introduction

With the emergence of the Semantic Web, it has been quickly recognized the desirability of applying its technologies to enhance the everyday desktop on which users work everyday. By exploiting the semantically richer (meta)data available for relevant items, it should be possible to provide more useful ways to access and process information stored in desktop items, such as files containing data, music, video, images; information about persons; information about events; etc…There have been several proposals on how to achieve this, notably Gnowsis (http://www.gnowsis.org) and Haystack (http://haystack.lcs.mit.edu/), among others.

The availability of metadata describing desktop items, especially using Semantic Web technologies, immediately suggests that it should be possible to access it as a hypermedia application, allowing the user to navigate from one item to its semantically related items. This paper shows how we have taken a development environment for hypermedia applications in the Semantic Web, called HyperDE, and generated a Semantic Desktop browser called HyperSD, which allows accessing desktop information in hypermedia fashion.

A brief summary of SHDM and HyperDE

The HyperDE environment allows the implementation of web applications designed using the SHDM method [1,2]. SHDM is a model-driven approach to design web applications using five different steps: Requirements Gathering, Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. Each phase

focuses on a particular aspect and produces artifacts detailing the application to be run on the web.

In SHDM we define a Semantic Web application as a navigational view over some ontology which describes the problem domain (see [4] for a more extensive discussion). We profit from being able to represent both data itself and its schema (meta-data) using the same formalism, since the fact that the schema can be manipulated just like any other kind of data brings greater expressiveness and conciseness to the specifications.

The HyperDE (http://server2.tecweb.inf.puc-rio.br:8000/projects/hyperde/trac.cgi/) environment is based on the MVC framework, and allows the designer to input SHDM navigational models (the "model" in the MVC framework), and interface definitions (the "view" in the MCV framework), and generates complete applications adherent to the specification. It also provides an interface to create and edit instance data, although, strictly speaking, this should actually be part of the generated application. In HyperDE. SHDM meta-models, user defined navigation models, as well as the application instance data, are all stored as RDF data.

The Semantic Desktop as a Hypermedia application

A Semantic Desktop can be seen as a meta-data enriched set of information items accessible through the user's desktop. Typical item are of different types, such as various file contents, such as text, mp3s, videos, programs, etc….; information about persons; and information about events. Fig. 1 shows a simple schema for these information types. It should be noted that the properties and some relations in this schema are the same as in popular existing ontologies for existing ontologies for similar concepts, e.g., FOAF for describing persons and relations, RDFCal for events, etc…

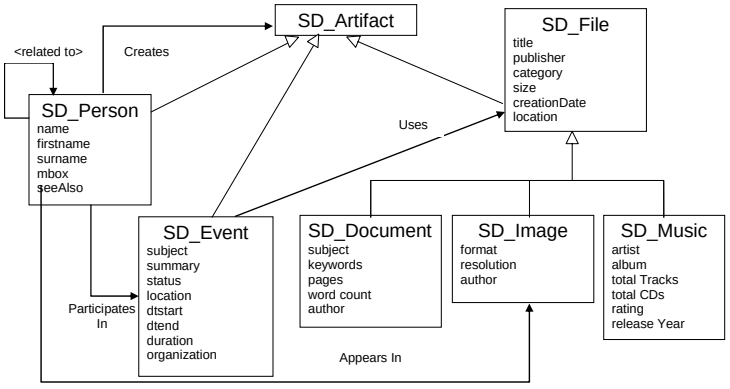


Fig. 1. A simple vocabulary for HyperSD.

Given the set of information items, a Semantic Desktop browser can be seen as an application that allows navigation among these items, as well as the ability to process them.

Following SHDM, we have defined a simple navigation schema for such an application, which was implemented in HyperDE; the resulting application is called HyperSD.

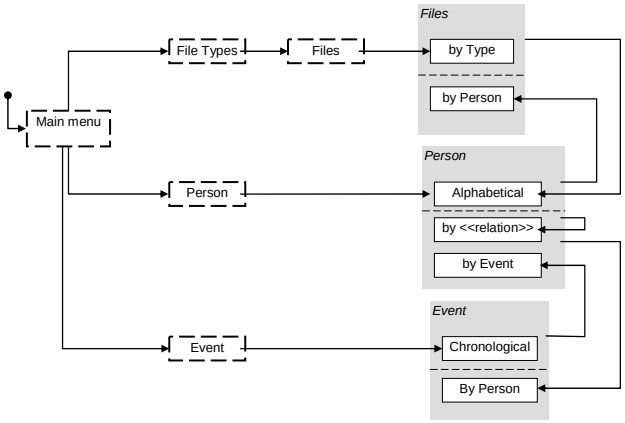


Fig. 2. Navigation Schema for HyperSD.

In Fig. 2, we show an initial navigation context diagram for HyperSD. In this diagram, dashed boxes are indices, and solid boxes are contexts (sets of items whose type is shown in the grey area). From the main menu, one can access a hierarchical index of files by file type, an alphabetical index of persons, and a chronological index of events. From a file, it is possible to navigate to the persons related to it. From a person, it is possible to navigate to the related persons according to each possible type of relation defined, to the files and to the events associated with that person. From an event, it is possible to navigate to the persons associated with this event.

It is important to stress that we do not claim that this is the only possible, relevant or meaningful way to navigate among items, but rather the fact that, having an environment such as HyperDE, it is very straightforward to implement whatever navigation topology is desired, and evolve it over time.

The Semantic Desktop implementation in HyperDE

Given the models described earlier, we implemented them in HyperDE, running as a web application local to the user machine. Fig. 3 shows the details of an Event in the context “Events in Chronological Order”. Clicking on a attendee of the event, one navigates to the person’s details, shown in Fig. 4.

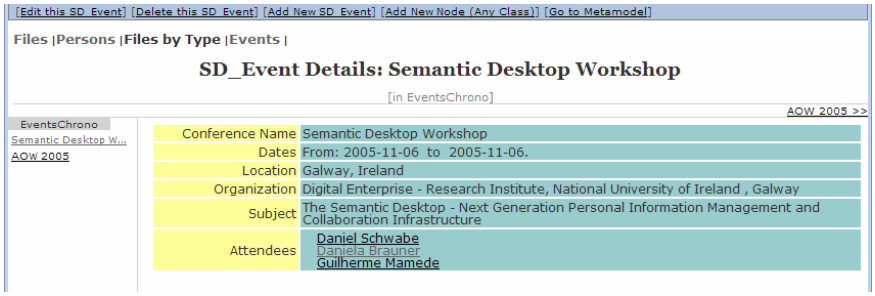


Fig. 3. An event (in the context Events in Chronological Order), in HyperSD.

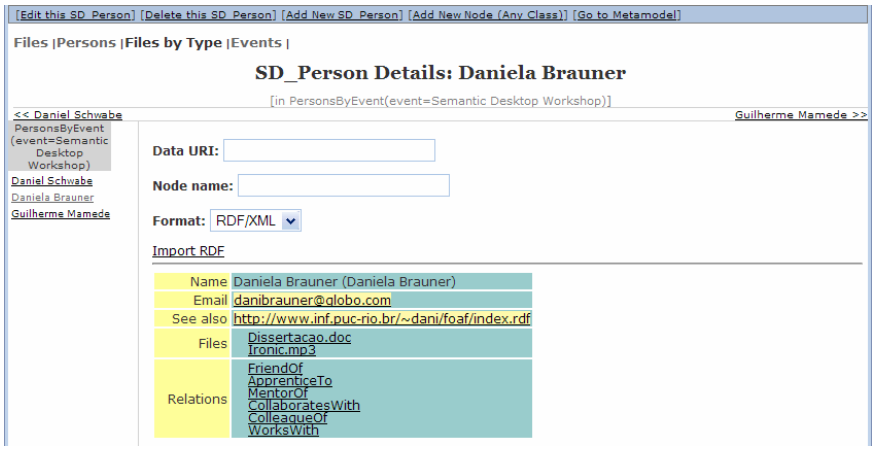


Fig. 4. Details of a Person in HyperSD.

The HyperDE environment provides contextual navigation automatically, according to the model specification. In this case, there is an index to all elements, shown in the left column of the screen, and sequential navigation in the Persons by Event, which generates ">>" anchor, next to the name of the next item in the context. Notice, in the example, that it is also possible to import a FOAF description to instantiate a new SD_Person. Notice also that there is an index of Relation Types (e.g., FriendOf, ApprenticeTo, etc…) associated with a person; this index is generated by a meta-query (i.e., a query to the model definition) extracting all defined relation types between SD_Person and itself.

HyperDE has a full fledged view definition mechanism allowing the customization of presentation templates, although it also possible to use a default presentation template that is built into the environment itself. In addition to defining views, the HyperDE enviroment also has facilities for the definition of all SHDM primitives (navigational classes, indexes, links, contexts). Fig. 5 shows an example of the interface for defining a navigation context, "Persons by Event", which retrieves all triples of type "SD_Person" that "Attends" an event which is passed as a parameter (the "?" in the query). Notice that we have extended RQL with an "Order by" clause.

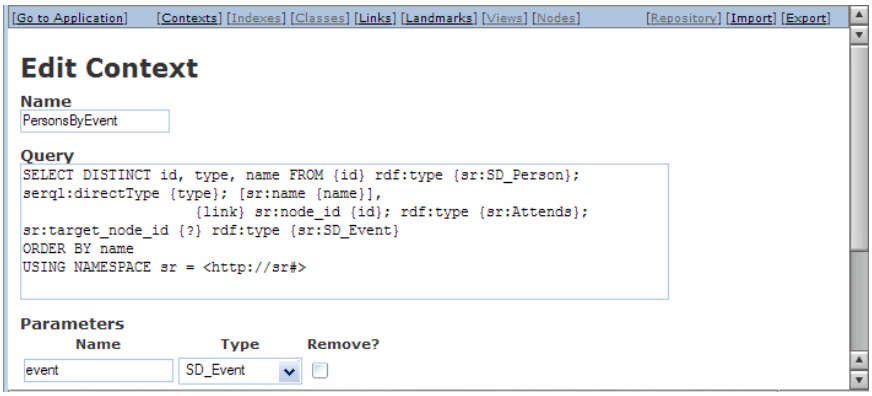


Fig. 5. The definition page in the model editor, for context Persons by Event.

Conclusions

We have defined and implemented HyperSD as a Semantic Web application that manipulates information items typical in a desktop, using the HyperDE environment. Current and future work being pursued includes

- Developing a richer set of wrappers;
- Providing a more sophisticated navigation structure, include also the possibility of allowing the user to customize it;
- Providing a more amicable interface look and feel;
- HyperSD currently runs as a local web application. We are investigating ways of integrating distributed HyperSD, running in individual machines

Acknowledgements. Daniel Schwabe is partially supported by a grant from CNPq, Brazil.

References

1. Lima, F.; Schwabe, D.: Modeling Applications for the Semantic Web. Proceedings of the 3rd Int. Conference on Web Engineering (ICWE 2003), Oviedo, Spain, July 2003. LNCS 2722, Springer Verlag, Heidelberg, 2003. pp 417-426. ISBN 3-540-40522-4.
2. Lima, F.; Schwabe, D.: Application Modeling for the Semantic Web. Proceedings of LA-Web 2003, Santiago, Chile, Nov. 2003. IEEE Press, pp. 93-102, ISBN (available at http://www.la-web.org).
3. Nunes, D.; "HyperDE - a Framework and an Ontology-driven Development Environment for Hypermedia Applications", MSc Thesis, Dept. of Informatics, PUC-Rio, Brazil, Feb. 2005.
4. Rossi, G., Schwabe, D. and Lyardet, F.: Web Application Models Are More than Conceptual Models. Proceedings of the ER'99, Paris, France, November 1999, Springer, 239-252.

Keywords and RDF Fragments: Integrating Metadata and Full-Text Search in Beagle++

Tereza Iofciu, Christian Kohlschütter, Wolfgang Nejdl, Raluca Paiu

L3S Research Center / University of Hanover
Deutscher Pavillon, Expo Plaza 1
30539 Hanover, Germany
{iofciu,kohlschuetter,nejdl,paiu}@l3s.de

1 Introduction

Full-text search engines and metadata repositories have so far investigated very different approaches to search, mainly due to their separate and different storage systems for information and data. As we have argued in previous papers, though, integrating full-text and metadata search capabilities is crucial for powerful semantic desktop search systems [3]. Semantic metadata is able to represent important contextual information on the desktop and provide both more research results (better recall) as well as more sophisticated ranking (better precision) compared to simple full-text search.

Recently, integrating databases and information retrieval technologies has been discussed by different research groups. [1] motivates the need for systems which integrate DB and IR querying capabilities, especially in the case of searching in libraries, where librarians are forced to switch between full-text and database search functionalities. An interesting approach to querying semi-structured data such as XML has been presented in [2], which introduces the notion of XML fragments to be used as parts of queries, inspiring part of the work described in the current paper. A system building on RDF is Magnet, which is presented in [4]. It supports naïve-user navigation of structured information via a domain-independent search framework and user interface. Magnet can be used both for browsing RDF as well as XML collections of data.

In our context we want to achieve integrated full-text and metadata search on the desktop including two important functionalities present in current search engines or IR systems, namely the possibility to return results which only partially match the query and to rank results accordingly. In contrast to this, usual database queries return all results as an unranked set of hits.

2 Searching with RDF Fragments

2.1 The Vector Space Model

For conducting full-text search, it is common to use the well-known Vector Space Model (VSM). An important aspect of this model is that documents and queries

have the same structure. This means that, when searching full-text document collections, the queries are *compared to full-text documents, just as two documents are.* The results are then ranked according to the *similarity* of the documents with the query. The relevance of a document d for a specific query q is computed by using a similarity function such as the cosine measure:

$$\rho(q,d) = \frac{\sum_{t \in q \cap d} w_q(t) * w_d(t)}{\|q\| * \|d\|} \quad (1)$$

The similarity $\rho(q,d)$ is computed in a t-dimensional space, t being the number of different terms appearing in all documents. $w_x(t)$ stands for the weight of term t in x, x representing the document or the query. This can be boolean (0 = not in document, 1 = contained) or, for instance, defined by the TF·IDF measure. The latter normalizes the impact of frequent/infrequent words in a document per division of a term's occurrences in a specific document by the number of overall documents the term is contained in. Looking closer at this similarity function, the reader can notice that results do not necessarily have to include all query terms. If they do not include at least one, though, the similarity will be zero. In contrast to the Euclidean distance, in the cosine similarity function only those terms influence the similarity value which occur in *both* vectors.

2.2 Searching both Full-Text and Metadata

How do we generalize the vector space model to a semantic desktop search scenario? The key idea here is to consider extended documents which consist both of full-text and of metadata, and allow the user query for these documents in an integrated way. Figure 1 shows such an extended document, with an RDF graph representing the metadata associated to the document.

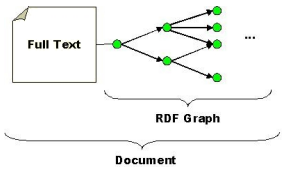


Fig. 1. A searchable document, consisting of full-text and reachable metadata

A query containing full-text and RDF metadata properties can then be expressed in the same way as a small document, including both keywords and RDF fragments. This approach offers a very easy-to-use query language. For example, if we consider the natural-language query: *"Return the papers which I received from Bob via an email attachment, considering RDF indexing"*, this can be expressed by the following RDF fragment query:

"`RDF indexing`" */storedFrom/attachedTo/receivedFrom* `Bob`.

If Alice remembers that the paper Bob sent to her has been presented at a VLDB conference, she can use this additional information to refine her query. So, the query now is *"Return the papers which I received from Bob via email, considering RDF indexing, and that have been presented at VLDB"*, which is represented as an RDF fragment query as follows:

"`RDF indexing`" */storedFrom/attachedTo/receivedFrom* `Bob`.
/presentedAt `VLDB`.

It is useful to support approximate and imprecise metadata queries as well. For example, in the previous query, Alice could have erred on the name of the conference. So articles presented at SIGIR, which matched the keyword and *receivedFrom* restrictions, should be also returned as hits, though with a smaller rank. In addition, we should also return appropriate documents if a metadata path has only been specified incompletely (for example, */receivedFrom* `Bob`).

2.3 Semantics of RDF Fragment Queries

Besides providing an intuitive and integrated way to query for full-text and metadata, we also want to be able to efficiently support indexing and searching on the desktop using a single VSM-based full-text search engine like Lucene. Based on the vector space model, we will then consider a document a potential result if it has a non-null similarity with the corresponding query. In addition to the basic vector space model functionalities, we also support additional restriction operators commonly used in full-text search engines: *MUST*, *MUST NOT* and *PHRASE* operators. Just like any other term, a phrase can be prefixed with the *MUST* or *MUST NOT* operator sign (`+` or `-`), to force its appearance/absence, or be enclosed in double quotes for the PHRASE operator to force a list of keywords to appear consecutively. If a term/phrase is not signed, its appearance is desired but not inevitable (i.e. an implicit *SHOULD* operator).

If we compare these query capabilities to relational query languages, our query model supports the operators selection and projection, as well as union, intersection and set difference. The only operator not supported is the join, which however only precludes queries such as 'Find papers from authors who cite themselves', seldomly thought of by search engine users.

3 Indexing and Querying of Full-text and Metadata

3.1 Background: Lucene and Beagle Indexing

We are building our search infrastructure on Beagle which is a GNOME project for indexing and searching resources on the desktop. Beagle uses Lucene as high performance full-text search engine, which is a multi-purpose information

retrieval library for adding indexing and search capabilities to various applications. Lucene can index and search any data that can be converted to a textual representation.

At the core of all full-text search engines is high performance indexing, i.e. processing the original data into highly efficient cross-reference lookup tables in order to facilitate rapid searching. A common way of implementing the vector space model is to store terms in inverted indices, associating them with the documents they are contained in as well as including the corresponding TF and IDF values.

3.2 Indexing Metadata to Support RDF Fragment Queries

The default approach to index metadata as used in Beagle is to define one field per metadata predicate; full-text and document URIs are also stored as fields. This idea is not suitable for metadata paths. Adding one field per path is not possible as the paths can contain a lot of different predicates, determined only at runtime. Furthermore, the ability to rank documents by metadata literals is lost because the TF·IDF measure used by the vector space model does not span across fields.

Our approach uses one single "metadata" field for all metadata associated to a document. For each directly associated statement, we store both predicate and object of the statement as text in this field. This makes storage independent of any underlying RDF schema, as the predicates are represented as terms rather than field names. To describe more complex contexts, we store *predicate paths*. These paths represent the properties which are not directly annotated to the document, but can be reached by following a path in the RDF graph starting from the document node via subject–predicate–object connections. RDF fragments can be matched using phrase queries on the metadata field, and results are returned almost instantaneously, as querying reduces to a lookup in the Lucene index.

4 Experiments

Our test dataset consisted of 150 publications, with metadata extracted from the CiteSeer database and some additional one regarding email information. We considered a community of three people, Alice, who has stored and indexed her publications on her desktop, and Bob and Chris from whom she received some of them. We have created additional annotations referring to the affiliation of some authors as well as regarding the source of the documents. We assumed 10 documents received as email attachments from Chris and 21 received from Bob. Publications are related to each other through citation, and are linked to persons by authorship or source relations. Paths are materialized up to literals, but not including other documents.

In our first experiment we searched for a document about *Web Communities*. If we search with this phrase only in the full-text, we get as hits the documents

119, 61 and 41 in this order and with similar rankings. If we add the title restrictions to the query we get 61, 119 and 14 with relevant differences in the ranking values. If we specify that the metadata query is mandatory, with the *MUST* operator, we only get document 61, 'Self Organization and Identification of Web Communities'.

In the second experiment we search for the phrase "semantic web" in the text field and get 24 documents. If we want to get documents written by persons affiliated to MIT we can narrow the search area by adding the keyword "MIT" to the text query. This approach returns 13 documents. If we add the metadata query '*/creator/affiliatedTo* `MIT`.' or just '*/affiliatedTo* `MIT`.', we get only two documents.

In the third experiment we search for documents received from "Chris Conti" and we got 10 hits, then we searched for documents received from "Bob Doe" and we got 21 hits. When we did a cross search, for documents received from "Chris Doe" we obtained no hits, which is exactly what we wanted to achieve.

In the first experiment, explicit metadata search has a big impact on ranking. Without the metadata restriction, Lucene TF·IDF scores were between 0.04 and 0.07. After adding the metadata phrase to the query, the first document (the one containing the keywords in the title) had a ranking of 0.1471. The terms "web" and "communities" appear less in the text field than in the metadata field. Together with the fact, that the metadata field is much smaller than the full-text field, the ranking considers matches in the metadata field much more important than in the full-text field.

5 Conclusions & Current Work

This paper has presented an integrated approach to indexing and querying full-text and metadata information which is both intuitive for the user and efficient to implement. It achieves this by building on top of a common full-text search engine and by only slightly modifying the usual Google-like search interface to allow the specification of RDF fragments as part of the query. Additionally, our approach provides inexact matching of user queries and ranking of results, evaluating metadata matches the same way as full-text matches, with the ability to return results even for incomplete and over-constrained metadata queries.

References

1. S. Amer-Yahia, P. Case, T. Rölleke, J. Shanmugasundaram, and G. Weikum. Report on the DB/IR Panel at SIGMOD 2005. In *SIGMOD*, June 2005.
2. D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML Documents via XML fragments. In *SIGIR*, New York, USA, 2003.
3. S. Ghita W. Nejdl P. Chirita, R. Gavriloaie and R. Paiu. Activity Based Metadata for Semantic Desktop Search. In *ESWC*, Greece, May 2005.
4. V. Sinha and D. R. Karger. Magnet: Supporting Navigation in Semistructured Data Environments. In *SIGMOD*, June 2005.

Pen-based Acquisition of Real World Annotations for Semantic Desktops

Markus Stäudel[1], Bertin Klein[2], and Stefan Agne[2]

[1] Fachhochschule Kaiserslautern, Standort Zweibrücken
Amerikastr. 1, 66482 Zweibrücken, Germany
mast0005@student-zw.fh-kl.de
[2] German Research Center for Artificial Intelligence (DFKI GmbH)
Postfach 20 80, 67608 Kaiserslautern, Germany
{bertin.klein,stefan.agne}@dfki.de

Abstract. We explain our strategy towards a Semantic Desktop and our research and prototype. In this paper we complement these with the paradigm of paper — scribbling, annotating, revising—, an information interface for knowledge work, which has been well-tried and elaborated over the centuries. Everybody is highly trained for it. The combination can instantly improve the conditions of knowledge work. By removing one more felt barrier, it has the potential to enhance a phenomenon called flow, providing users with efficiency and satisfaction at work. The implementation is built around a pen transmitting its movements and a state-of-the-art handwriting recognition.

1 Introduction

The regular (wooden) desktop seems to us the authoritative archetype for any semantic desktop. Semantics on the regular desktop is found in structures that are meaningful to users, and in their creation, change, and maintenance ([15], compare also [2, 4]). Our approach to new breeds of semantic desktops starts at the concepts and procedures on regular desktops. [6, 16] Further, our foremost goal is to reduce the users cognitive load and his mental distraction, because a mind relieved from distractions, draws the best results from the sensed input and his prior knowledge. [19] Thus we try to sensitively elicit the real tasks and distractions[3] of users and quietly assist these (cf. [4, 9]).

Notwithstanding major achievements in personal computing, reading and writing demand for paper. In a study aimed at knowledge capturing needs and uses at desktops Adler et al [1] conclude among other interesting things that: "Reading occurs more frequently in conjunction with writing than it does in isolation. Thus, it appears that writing (in a variety of forms) is an integral part of workrelated reading. Designers need to seriously consider the importance of developing reading devices which also support the marking or writing of documents during the reading process." [1] Other work emphasizes the impact of the reading-writing combination on creativity. [13, 14] Notwithstanding, that creativity is a well-understood issue lately, there is also a much more

[3] Any minimal, intermediate, even unconscious goal or indirection which requires a microsecond of conscious attention.

profane view on the topic: even on such simple tasks as dragging and moving, users perform better with a stylus (on some canvas) than with a trackball or a mouse. [12]

We conclude that users should be enabled to print and work with paper. TabletPCs are an approach, however, with the same OCR software and only 40 Euros of hardware, we additionally get the high resolution of paper and the freedom to skim over and spread pages. While scribbling and annotating, the strokes are captured and sent to the electronic desktop. Plug-in modules can edit the electronic version of the document. The plug-in module actually first completed, is a Microsoft handwriting recognition, the output of which is used to automatically place comments at any position into the original electronic document.

2 The Whole Approach: Semantics, Users, Desktops

Writing desks exist for two reasons. First, they are needed as a space for documents and reminder notes, while forging ahead in constructing their semantics, as long as their meaning has not been fully incorporated into the mind of the user. Second, computer desktops lack important, useful elements. — Supporting users in the first by improving the latter, we have the goal to deliver *flow*: It happens that one starts a day in the office at ones writing desk and suddenly one kind of wakes up, and it is two hours later. One has achieved a lot, but cannot easily tell what in detail one has done. This phenomenon is called ”flow”. It enhances individual work, especially where creativity is involved, and yields to the efficiency and satisfaction of individuals in their work. [3]

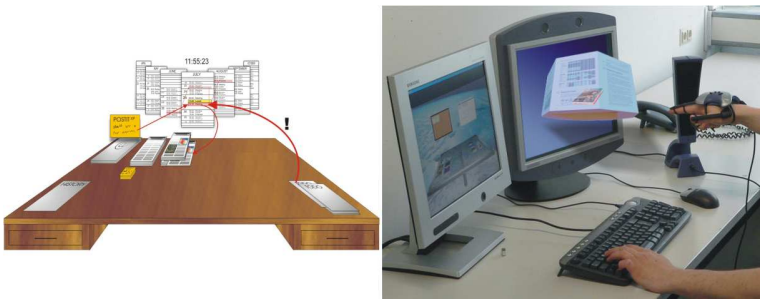


Fig. 1. A regular writing desk it the archetype of a Semantic Desktop. The right side conveys an idea of the visualization elements (comprising 3D) with which we are experimenting.

2.1 In Support of the Semantics of a Document Collection

With books, folders, reports, notes, a calendar, todo-lists —i.e. a collection of documents— people organize their knowledge or information, in order to fulfill their daily work. Semantics here is that what such collection of documents mean to their owner, comprising the topics, statements, conclusions, interrelations, and implications discovered by the

owner in these documents. We like the iterative flavor of “Keeping found things found” from the University of Washington, which we consider an important aspect of the so-called *hermeneutic circle*, i.e. the laborious and time consuming integration of every new document into the other knowledge known by a person. The expert might notice, that we conceive of semantics for a user as something like RDF or equally Topic Maps, in the brain of the user. [5]

Our implementation and research is based on the idea that thinking involves the senses, e.g. the visual. Figure 1 on the left shows a design sketch of our semantic desktop. On the right it shows a real screen and a mockup of the (really running) 3D screen. The left image comprises not yet completely implemented visual elements like document viewers, document stacks, Post-Its, meta-information (links), and a calendar tool with enhanced fisheye view, and active links to documents. The right side shows our visualization equipment, trying to tap the 3rd dimension. The monitor pair is actually used like that, as it was found to be more efficient to complement 3D viewing with a regular 2D view. We also experiment with a powerwall.

2.2 Virtual Desktops are Lightweight and Semantic

With all my paper documents from my writing desk as PDF documents on a USB stick I can carry them with me at all times. A near future smart USB stick will be pluggable to displays — perhaps a futuristic projector or my stylish glasses — and allow me to spend a few seconds or longer at my writing desk. Note, that all my todo-lists, my calendar, my notes, my documents, are there, my entire writing desk.

The instant advantage of a virtual, semantic desktop can be sketched with electronic “Post-Its”: Electronic “Post-Its” can be retrieved, e.g., by its creation time, “this morning”, and with a mouse click one finds the document to which it refers. “Post-Its” can also refer to several documents and thus establish commented links. They can also include links to appointments, todo-lists, etc. It is also possible to let popup a required document at a specific time and at a desired text position depending on automatically extracted information about the document. Paper “Post-Its”, on the other hand, lack these features.

3 Gaining from Paper Editing

Imagine I am asked to do a review of an article, enclosed as a PDF. I print it, put the paper into a clamp and start commenting. The pen transmits its movements to the clamp. On my screen I have opened the PDF version of the document and all the comments I write on the paper appear synchronously in the PDF, nicely in Times Font. When I physically turn a page I tip into the lower right corner of the page, and the editor turns page too. When I finish, I just save the document and mail it back.

Paper is an input catalyst. Many projects to overcome the boundedness of paper failed miserably. [15] Field research has elicited that a couple of tasks are done exclusively or with high tendency with paper, like revising another’s person text [15], collaborative authoring, telephone activities or at planning and thinking. [16] It is possible to navigate quickly, apply annotations, underline important words or passages,

read through again and the like. This is not comparably possible on today computer desktops. [13] Sure, only as an exception it is easier to actually edit with paper.

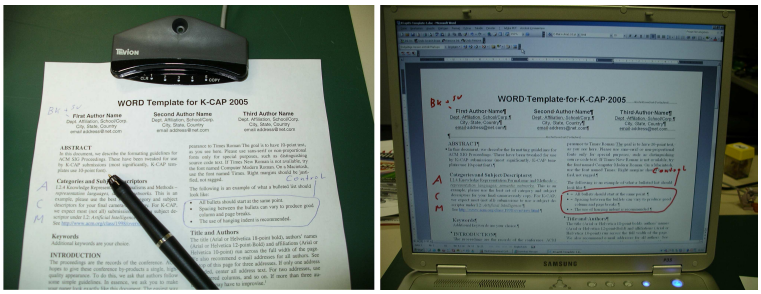


Fig. 2. Annotating a printed dokument and the result on the screen.

A stylus, a clamp, and our Gesture Port can do the job. The 40 Euro of hardware needed comprises a stylus and a clamp. The stylus is equipped with a transmitter, which sends coordinates (plus some events like PENUP, PENDOWN, PENMOVE, ...) wirelessly to a receiver embedded in the clamp and further to the desktop software (usually vie USB). The Gesture Port interfaces input devices to our Desktop framework. An intermediate unit converts the movements received into meaningful gestures. Here, we started with a (rather good) pre-release Microsoft handwriting recognition. Further, we committed (without loss of generality of the approach) to a PDF editor, as the (fixed) target application.

A special set of stylus gestures changes the interpretation mode, so that the interpretation unit then sends commands to the target or changes the target. This allows to serve different stylus based activities, as drawing (movements cause lines to appear), text editing (characters are recognized and inserted, striking through deletes text), text formatting (underlining, italicizing, ...), layout editing (paragraphs are cut, merged, moved ...), annotating (textual annotations and links are edited/inserted, the visible document text remains unaltered). Currently, only positioning and editing of comments are wholly implemented. In general, the Gesture Port can also interface the data glove visible in Figure 1. However, for writing we found the data glove rather inconvenient.

4 Related Work

Recently, there was *the* assortment of related work with respect to Semantic Desktops made meet under the label of “Personal Information Management”. [7] Certainly, www.gnowsis.org remains to be added. A couple of approaches are specificly relevant to our paper philosophy: DigitalDesk [20], VideoMosaic [10], Ariel [11], InteractiveDesk [17], EnhancedDesk [8], MetaDesk [18]. Most of these Desk Systems try to or actually use input devices like cameras and scanners. Some articulate goals very similar to ours, but the approaches differ astoundingly. E.g. instead of a working space

on the desk to read and write, displays are embedded into the surface. Our specific task of annotating documents is not addressed at all.

References

1. A. Adler, A. Gujar, B. Harrison, K. OHara, and A. Sellen. A diary study of work-related reading: Design implications for digital reading devices. In *Proceedings of CHI 98*, 1998.
2. T. Berners-Lee. Design issues – architectural and philosophical points, October 1998.
3. M. Csikszentmihalyi and I. Csikszentmihalyi. *Optimal Experience : Psychological Studies of Flow in Consciousness*. Cambridge University Press, 1988.
4. M. Eldridge, A. Sellen, and D. Bekerian. Memory problems at work: Their range frequency, and severity. Technical Report EPC-1992-129, RXRC Cambridge Technical Report, 1992.
5. V. Gallese and T. Metzinger. Motor ontology: The representational reality of goals, actions, and selves. *Philosophical Psychology*, 13(3), 2003.
6. R. Harper, A. Munro, H. Hook, and D. Benyon. Information that counts: A sociological view of information navigation. In *Social Navigation of Information Space*, 1999.
7. W. Jones, H. Bruce, N. Belkin, V. Bellotti, S. Dumais, J. Grudin, J. Gwizdka, A. Halevy, D. Karger, D. Levy, M. Perez-Quinones, and J. Raskin. The pim workshop — an nsf-sponsored invitational workshop on personal information management, January 2005.
8. H. Koike, Y. Sato, and Y. Kobayashi. Integrating paper and digital information on enhaced-desk: a method for real-time finger tracking on augmented desk system. *ACM Trans. on Computer-Human Interaction*, 8, 2001.
9. M. Lamming, P. Brown, K. Carter, M. Eldridge, M. Flynn, G. Louie, P. Robinson, and A. Sellen. The design of a human memory prosthesis. *Computer Journal*, Vol. 37(3), 1994.
10. W. Mackay. Augmented reality: linking real and virtual worlds. In *In Proceedings of ACM Conference on Advanced Visual Interfaces, AVI '98*, 1998.
11. W. Mackay, D. S. Pagani, L. Faber, B. Inwood, P. Launiainen, L. Brenta, and V. Pouzol. Ariel: augmenting paper engineering drawings. In *CHI '95: Conference companion on Human factors in computing systems*, 1995.
12. I. S. MacKenzie, A. J. Sellen, and W. Buxton. A comparison of input devices in elemental pointing and dragging tasks. In *Proceedings of SIGCHI 91*, 1991.
13. K. O'Hara and A. Sellen. A comparison of reading on-line and paper documents. In *Proceedings of CHI 97*, 1997.
14. K. O'Hara, A. Taylor, W. Newman, and A. Sellen. Understanding the materiality of writing from multiple sources. *International Journal of Human-Computer Studies*, Vol. 56(3), 2002.
15. A. Sellen and R. . Harper. *The Myth of the Paperless Office*. MIT Press, 2003.
16. A. Sellen and R. Harper. Paper as an analytic resource for the design of new technologies. In *Proceedings of CHI 97*, 1997.
17. A. Toshifumi, M. Kimiyoshi, K. Soshiro, and H. Shojima. Interactivedesk: a computer-augmented desk which responds to operations on real objects. In *Conference companion on Human factors in computing systems*, 1995.
18. B. Ullmer and H. Ishii. The metadesk: models and prototypes for tangible user interfaces. In *Proceedings of UIST '97*, 1997.
19. J. Waterworth and E. Waterworth. Affective creative spaces: the interactive tent and the illusion of being. In *Proc. of The Int. Conf. on Affective Human Factors Design*, 2001.
20. P. Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7), 1993.

Semantic Pen - A Personal Information Management System for Pen Based Devices (Extended Abstract)

Akila Varadarajan, Nilesh Patel and William Grosky

The University of Michigan - Dearborn, Dept. of Computer Science,
4901, Evergreen Road, Dearborn, MI 48080, USA
{akilav, patelnv, wgrosky }@umich.edu

Abstract. The Onset of Semantic Web technology have promised a new vision of Personal Information Management (PIM). With the advent of Pen-based computing, PIM faces new challenges: usability and flexibility are important constraints in the pen based environment. We present our system of Semantic Pen - an augmented pen based PIM system that merges the efficiency of semantic web with the usability of pen based devices. The architecture consists of an intuitive user interface which can capture digital ink, a Hidden Markov model (HMM) to extract personal information and a data model of Resource Description Framework(RDF) for flexible organization and semantic querying of data.

1 Introduction

Personal Information Managers (PIM) have become increasingly common these days. The usage model of PIM systems have gone beyond scheduling reminders and simple record maintenance. Semantic Web, through the introduction of ontological reasoning by means of Resource Description Framework(RDF)[1] have proven to be an efficient solution for PIM . The Haystack Project [2] is well known for applying semantic web technologies to create a fully flexible and customizable PIM portal for organizing the germane information. The Gnowsis Semantic desktop [3] targets data integration including data from 3rd party applications. Semex[4] focuses on personalized desktop search. Chandler[5] is an Interpersonal Information Manager that supports data sharing besides managing email, calendar and other general information. Retsina Calendar Agent[6], is a distributed meeting scheduling agent which works in conjunction with Microsoft Outlook 2000 and Semantic Web.

While most of the research in PIM using Semantic Web is centered around desktop and notebooks, there is a need to extend such concepts in context of pen-based computing. The pen-based systems have empowered users by providing the most natural form of input modality known as *Digital Ink*. Since its introduction, researchers have shown increased interest to ease the user interface centric tasks. Wilcox et al. designed a system Dynomite [7] for organizing telephone numbers and other tasks by applying properties for ink words. Scribbler [8] is another tool

that enables searching ink words, symbols or simple sketches by matching raw strokes instead of recognized text. Marquee [9]is a logging tool where users can correlate their personal notes and keywords with a videotape during recording. Microsoft's products One Note 2003 and Journal helps to capture, customize and organize ink documents suitably.

We present *Semantic Pen* that aims to combine the efficacy of semantic web with the usability of pen based devices to provide a next generation highly intuitive and intelligent PIM system.

2 Semantic Pen

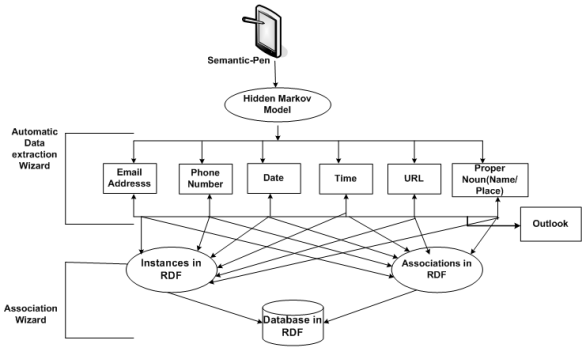


Fig. 1. Architecture of Semantic-Pen

Semantic Pen has a simple and attractive user interface comparable to leading note taking tools. In addition, our system is composed of two core modules; (1) an Automatic Data Extraction (ADE) wizard and (2) an Association wizard. The system Architecture of Semantic Pen is shown in figure.1. ADE is the heart of the system which extracts the data via Hidden Markov Models(HMM)[10]. Additional details such as name of a person for an extracted email address can be semi-automatically included through the ADE wizard. This wizard displays a *smart name/place list* generated by our intelligent noun filter algorithm. The user can either choose a name from the list or enter his own. This extracted personal information is then automatically stored in the commercial information management tool such as Microsoft Outlook. Once the personal information is extracted, an *Association Wizard* helps associating the data with the existing data repository items. Our approach uses the popular RDF framework Jena [11] to store and retrieve the data.

2.1 Automatic Information Extraction using Hidden Markov Model(HMM)

HMM is a finite state automation that implements stochastic state transitions and symbol emissions. We use the model of Freitag and MacCallum [12, 10] to extract personal data from the ink notes. Once the states for the HMM (Prefix, Target,Suffix and Background states) is decided, the document is parsed and taxonomized to obtain the emission vocabulary of the HMM. We generate a set of intuitive term by feature pairs t, f where t is the intuitive term and f is an identified feature that creates the appropriate intuition on that term [13].

The possible formats and constraints for the $Intuitive\ term\ features$ such as *Email ID*, *Phone No*, *Date*, *Proper Noun* are identified and defined in a database. Then we calculate WFM to classify the intuitive terms. For a term t, $WMF(t)$ is computed as follows:

$$WFM(t) = \frac{Nc(t)}{Nc(f)}$$

Where, $Nc(f)$ represents the total number of constraints for the word feature f. For example, the '@ 'symbol and a domain name are some constraints for an email address. $Nc(t)$ is defined as:

$$\sum_{x=0}^{Nc(f)} M(t, c_x)$$

where $M(t, c_x)$, the matching function, equals 1 if the term t contains a matching constraint c_x

$WFM(t)$ is calculated by varying f in $Nc(f)$. If $WFM(t)$ equals 1 for some value of f in $Nc(f)$, it means the term t is of the suspected word feature type f. If $WFM(t)$ is less than 1 for all values of f in $Nc(f)$, it means the term is not of any type of suspected word feature.

Table.1 describes how we define the emission vocabulary for the HMM by means of WFM and Bikel's classification of word features [13].

Table 1. Emission vocabulary for HMM

Intuitive Word Feature	Example formats
Email ID	bob@umich.edu, bob@yahoo.com, bob@xyz.org
Phone No.	(586)-779-6320, 586-779-6320, 779-6320
Date	09/01/06, 09-01-06, 09/01/2006,Sep-1-06
Time	12.30 pm, 12:30 a.m, 12.30 AM
Proper Noun(Name or Place)	Bob, Michigan
URL	www.umich.edu

Once the emission vocabulary by means of the intuitive word features is obtained, the Viterbi algorithm [12] is used to accurately identify the most likely state sequences of a particular document. Finally, the HMM outputs the strings which are likely to be the personal data that need to be stored.

2.2 Personal Information Association using RDF

The next step is to create suitable associations of the new data with the existing elements in the database. We are currently in the development stage of this algorithm. In this, we define two components namely *instances* and *associations*. The instances are the actual objects that need to be associated such as *email address of Bob* or *web page of an institution "XYZ"*. The associations are the relationship that might exist between two instances. Consider, "Bob *works at* XYZ". In this case *works at* is an association that exists between instances Bob and XYZ that binds them together. Similarly there might be another association existing such as "Steve *works at* XYZ". Now an automatic link gets associated between Bob and Steve. However, the user is prompted to obtain a suitable association between these two instances.

Initially all possible instances such as contact information, task list and web page links will be extracted by the system. Associations among these instances will be obtained semi-automatically by running the *association wizard*. The instances and associations are then stored in a separate database and represented by means of Ontolgies using Resource Description Framework (RDF). The RDF framework Jena [11] is chosen to store and retrieve the RDF data. Also, when a new item is added to the database externally, our system will alert the user to run the *association wizard* to form suitable associations.

Our interface will identify the associated instances by querying the RDF database and generate associations such as;(i) a calendar entry is related with a file which is modified at that date and time,(ii) a book marked web page consists of information about a workshop in the task item, (iii) a contact is the author of a particular document. The user will be allowed to choose an association from an existing list or to define his own.

3 Experimental Results

A NEC Versa Lite Pad Tablet was used to test our system. The note taking interface is developed using Agilix infinotes [14] .NET component. To test our Automatic Data Extraction (ADE) wizard, we collected meeting notes from 25 people. Each collected note was about 250-500 words in length, containing a mixture of email address, phone number, date-time information, proper nouns, and hyper-links. The data extraction results were analyzed off-line via the *Automatic Data Extraction(ADE)* wizard. Our application uses the recognized ASCII text from the meeting notes for all manipulations. The built-in Microsoft Hand Writing Recognizer that comes with the tablet is used to translate ink data to

the ASCII text. Since the *Association Wizard* is still under development, the experimental results pertaining to only ADE is presented in this paper.

The ink to text recognition accuracy plays a major role in performance of ADE wizard. In past, due to its least individuality the numbers have been reported with higher recognition accuracy [15]. Our analysis also supports the previous research in this regard. The recognition rate of numbers in our experiment, was found to be as good as 88.9% compared to the recognition rate of the letters which was found to be just 58.8%. Similarly, we also found that the non-cursive handwriting had the highest recognition rate of 82.2% compared to the cursive handwriting recognition rate of about 73.2%. The printed handwriting had the worst recognition rate of 21.1%.

In addition to the recognizer's inaccuracy, we also found that the emission vocabulary symbols failed to get identified due to unnecessary white spaces inserted by the recognizer. Our system intelligently handles these white spaces to improve an overall precision of the data extraction system.

The results of our phase I activity in extracting key personal information using HMM is measured using standard precision and recall measures, defined as:

$Precision = \frac{R}{R+R_i} * 100$; $Recall = \frac{R}{R+R_m} * 100$

where, R=Relevant records retrieved, R_i= Irrelevant records retrieved and R_m=Missed relevant records.

Table 2. Precision Vs Recall for automatic data extraction

Data Extracted	Precision	Recall
Email address	90.15	89.34
Phone Number	96.57	96.87
Schedule Information (Date and Time)	88.26	89.75
URL	91.12	92.34
Proper Noun(Name or place)	93.23	89.23

References

1. Guha, R., Brickley., D.: Rdf vocabulary description language 1.0: Rdf schema. W3C recommendation 10 february 2004. (2004)
2. Quan, D., Huynh, D., Karger, D.R.: Haystack: A platform for authoring end user semantic web applications. (ISWC 2003) 738753
3. Sauermann, L.: The gnowsis semantic desktop for information integration. (Proceedings of WM 2005)
4. Cai, Y., Dong, X.L., Halevy, A., Liu, J.M., Madhavan, J.: Personal information management with semex, Baltimore, Maryland USA, ACM (2005)
5. OSAF: Chandler. "http://www.osafoundation.org/Chandler_Compellin_Vision.htm" (2004)

6. Payne, T.R., Singh, R., Sycara, K.: Calendar agents on the semantic web. IEEE INTELLIGENT SYSTEMS. (2002) 84–86
7. D.Wilcox, L., N.Schilit, B., Sawhney, N.: Dynomite: A dynamically organized ink and audio notebook. SIGCHI 1997, (ACM) 186–193
8. poon, A., Weber, K., Cass, T.: Scribbler: A tool for searching digital ink. CHI '95, ACM Press (1995) 252–253
9. Weber, K., Poon, A.: Marquee: a tool for real-time video logging. SIGCHI' 94, ACM (1994) 58–64
10. Taghva, K., Coombs, J., Pereda, R., Nartker, T.: Address extraction using hidden markov models. IS&TSPIE 2005 (January 2005)
11. Jena: A semantic web framework for java. (http://jena.sourceforge.net)
12. Freitag, D., McCallum, A.: Information extraction with hmm structures learning by stochastic optimization. 17th National Conference AI (2000) 584–589
13. Bikel, D., Miller, S., , Weischedel, R.: Nymble: a high-performance learning name-fnder. ANLP-97 (1997) 194–201
14. Agilix: Infinotes. (http://www.agilix.com/www/notecontrol.aspx?pid=14.)
15. Zhan, B., Sargur.N.Srihari, Lee, S.: Individuality of handwritten characters. ICDAR 2003, IEEE (2003)

Semantics-based Publication Management using RSS and FOAF

Peter Mika and **Michel Klein** and **Radu Serban**
Department of Computer Science, Vrije Universiteit Amsterdam
[pmika|mcaklein|serbanr]@cs.vu.nl

Abstract

Listing references to scientific publications on personal or group homepages is a common practice. Doing this in a consistent and structured manner either requires a lot of discipline or a centralized database. Scientific publication, however, is a distributed activity by nature. We present a completely distributed and RDF-based implementation for disseminating references to scientific publications. Our application only uses existing information sources and allows for different output formats, e.g. HTML, RSS and RDF.

1 Collecting and Publishing References

Information about scientific publications is often maintained by individual people. To present this distributed information in different selections and in different formats at different locations usually requires a lot of manual work. We demonstrate an application that performs this task using Semantic Web based techniques.

Our application collects several sources of information from several locations, in particular information about publications of authors from their homepage, information about group-membership from the department website and information about people by crawling FOAF-profiles. All sources are—if not yet in this format—translated to RDF and uploaded to an RDF store, in our case Sesame [Broekstra *et al.*, 2002].

In the repository we apply several unification and reasoning steps to link the different data sources, to derive additional facts and to remove redundant information. In addition, a separate web service can be used to query for publications based on specific criteria and to produce a variety of output formats, including BuRST (a compatible extension of RSS 1.0) and HTML.

Figure 1 presents a schematic representation of the approach, which is introduced in detail in the following section.

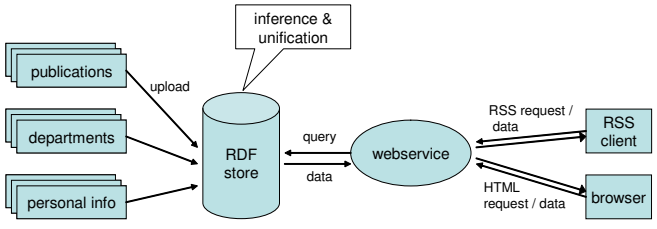


Figure 1: A schematic representation of the approach.

2 Sources of Information

For information about publications, we rely on the common BibTeX format. We ask authors to include a BibTeX file with their own publications on a publicly accessible part of their website. For many authors this does not require additional work, as they already maintain such a file themselves. A simple crawler collects all files from the `www.few.vu.nl` domain. The BibTeX files are translated to RDF using the BibTeX-2-RDF service,[1] which creates instance data for the "Semantic Web Research Community" (SWRC) ontology.[2]

Personal information is collected via the web as well, using the FOAF profiles [Brickley and Miller, 2005] that people linked from their homepage. The FOAF files contain RDF statements describing personal information such as the individual's homepage, workplace, image and relationships to other people.

To know which researchers are member of which department, we have implemented a web service that translates the content of the department mailing lists to a FOAF format with statements about group membership. We do not reveal the email addresses of people, but use a hash of the email address as identifier. By using the mailing lists as a source for the group membership information, we do not have to maintain this information ourselves, but rely on the existing infrastructure in the department (i.e. the computer system adminstration).

3 Aggregation

Mapping Schemas

Using distributed, web-based knowledge technologies, we have to deal with the arising semantic heterogeneity of our information sources. Heterogeneity effects both the schema and instance levels.

As the schemas used are stable, lightweight web ontologies, mappings on the class level cause little problem: such mappings are static and can be manually inserted into the knowledge base. An example of such a mapping is the subclass relationship between the `swrc:Person` and `foaf:Person` classes or the subproperty relationship between `swrc:name` and `foaf:name`.

Although we used existing RDF schemas for describing the instance data, a simple extension of the SWRC ontology was necessary to preserve the sequence of authors of publications. To this end we defined the `authorList` and `editorList` properties, which have `rdf:Seq` as range, comprising an ordered list of authors..

[1] See `http://www.cs.vu.nl/~mcaklein/bib2rdf/`.
[2] See `http://ontoware.org/projects/swrc/`.

Unifying Instances

Heterogeneity on the instance level arises from using different identifiers in the sources for denoting the same real world objects. This effects FOAF data (where typically each personal profile also contains partial descriptions of the friends), but also publication information, as the same author may be referenced in a number of BibTeX sources.

The solution is provided by instance reasoning (smushing) using ontological features. The FOAF ontology defines a number of inverse-functional properties of the Person class which can be used to determine whether two instances of Person are the same. (Functional properties, on the other hand, can be used to prove that two instances are not the same.) For example, if two Persons have the same value for the `mbox-sha1sum` (hash of the email address), we can conclude that both instances are the same. In this way, we can relate the statements from the FOAF files to the statements about the mailinglist-membership. Besides the inverse-functional properties, we also apply fuzzy string matching to compare person names, following a step of normalization (e.g. to be able to compare 'Harmelen, F.' and 'Frank van Harmelen'). Similarly, publications are matched based on an exact match of the date of the publication and a tight fuzzy match of the title. Matching publications based on author similarity is among the future work.

The matches that we find are recorded in the RDF store using the owl:sameAs property. Since Sesame doesn't natively support OWL semantics at the moment, we expanded the semantics of this single property using Sesame's custom rule language. These rules express the reflexive, symmetric and transitive nature of the property as well as the intended meaning, namely the equality of property values. The rules add several statements to give all the equivalent resources the same the set of properties. These rules are executed by the custom inferencer during uploads, which means that queries are fast to execute. (On the downside, the size of the repository greatly increases.)

4 Presentation

After the information has been merged, the triple store can be queried to produce publications lists according to a variety of criteria, including persons, groups and publication facets. An online form helps users to build such queries against the departmental publication repository. The queries are processed by another web-based component, the Publication webservice.

This tool takes the location of the repository, the query, the properties of the resulting RSS channel and optional style instructions as parameters. In a single step, it queries the repository and generates an RSS channel with the publications matching the query. This RSS channel follows the BuRST specification[3] for mixing in publication metadata into the RSS channel. The resulting channel appears as a RSS 1.0 channel for compatible tools while preserving RDF metadata.

The presentation service can also add XSL stylesheet information to the RSS feed, which allows to generate different HTML layouts (tables, short citation lists or longer descriptions with metadata). The HTML output can be viewed with any XSLT capable browser and it can be tailored even further by adding a custom CSS stylesheet.

[3] `http://www.cs.vu.nl/~pmika/research/burst/BuRST.html`

5 Use Cases

Our system for semantics-based bibliography management can be used by individuals and groups alike in a variety of modes. It can be used to provide a search interface to publication collections on personal homepages or departmental websites such as the homepages of the AI and BI groups of the VUA (information pull).

More interestingly, the use of RSS technology allows others to be notified of changes to these collections (information push) by subscribing to publication feeds. A number of generic tools are available for reading and aggregating RSS information, including browser extensions, online aggregators, news clients and desktop readers for a variety of platforms. While these software are not aware of the SWRC and FOAF schemas, they are still able to process BuRST feeds by ignoring the information they do not understand. (This behaviour is mandated by the RSS specification and is the basis of modularization in RSS.) Mozilla FireFox also natively supports RSS feeds as the basis for creating dynamic bookmark folders. These folders refresh their contents from an RSS feed whenever the user opens them.

The reliance on RDF and lightweight, widely used web ontologies also makes it possible to access personal profiles and publication information by generic RDF tools such as the Piggy Bank browser extension. Piggy Bank allows users to collect RDF statements linked to Web pages while browsing through the Web and to save them for later use. FOAF information can be processed by a growing number of tools, while the SWRC data can be easily converted back to BibTeX to complete the knowledge cycle.

6 Discussion

In summary, we presented a semantic-based system for publication management that builds on web technology, well-known ontologies and by reusing existing information requires no additional effort from the individual. In comparison to centralized approaches, our system leaves the control over publication management and presentation in the hands of the individual researcher, while still allowing for information push. On the other hand, our system is more lightweight than P2P networks that require users to install and run specific software on their computers. The Java object models for the FOAF, RSS and BuRST formats as well as the tools for crawling and smushing FOAF data have been made available as part of the open source Elmo API for Sesame. Elmo can be downloaded from `www.openrdf.org`. The interface to the tools themselves and some examples can be found at `http://prauw.cs.vu.nl:8080/burst/`.

References

[Brickley and Miller, 2005] Dan Brickley and Libby Miller. FOAF vocabulary specification. Namespace document, June 3, 2005.

[Broekstra *et al.*, 2002] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: An architecture for storing and querying RDF and RDF Schema. In Ian Horrocks and James A. Hendler, editors, *Proceedings of the First International Semantic Web Conference (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68, Sardinia, Italy, June, 9–12, 2002. Springer-Verlag.

Smarter Groups – Reasoning on Qualitative Information from Your Desktop

Sebastian Böhm, Marko Luther, and Matthias Wagner

DoCoMo Euro-Labs, Landsbergerstr. 312, 80687 Munich, Germany
<lastname>@docomolab-euro.com

Abstract. This paper explores the possibilities and core technologies of the ongoing development towards the Semantic Web for desktop application enhancement. It demonstrates how an ontology-based software can provide refined support for personal information organization. The Web Ontology Language and reasoning mechanisms based on Description Logics (DL) are analyzed as enabler technologies for semantic enrichments. We report on our prototype that realizes the enhancement of Apple's Address Book application by DL-based smart groups.

1 Introduction

The communication environment we find today dramatically changed the way we work and interact with other people. Access to common knowledge sources like the World Wide Web and means of communication like E-Mail have long become ubiquitous and intensified the need for more efficient and intelligent knowledge management mechanisms. In this respect, creating new approaches for an integrated information management and distribution is one key task the IT industry and the computer science community are currently dealing with. Therefore, a lot of effort is put into the development towards the Semantic Web [1] to provide a machine readable and meaningful description of the elements of the World Wide Web. Certainly, desktop applications could also profit exceedingly of well-founded logical annotations, which enable adequate techniques for handling distributed data more efficiently. Based on semantic descriptions, such applications would comprise additional reasoning mechanisms and therefore outmatch ordinary database driven approaches. In this context, we discuss an experimental approach for information management by establishing a link to technologies of the Semantic Web. To achieve this, a custom built set of ontologies as well as a prototype for ontology-based desktop application enhancement will be introduced.

2 Ontologies

Realizing interoperability between different knowledge representations, a set of core ontologies written in the Web Ontology Language (OWL) has been developed to model a concrete use case for application enhancement while at the same time providing a linkup to common upper context ontologies. The core ontologies altogether consist of nine components, defining more than 300 concepts and nearly 250 properties. Each of

these core ontologies contains a specific vocabulary concerning the representation of the domain of interest that can be used for further domain specific deployment.

The agent ontology is informed by the FOAF vocabulary [2] and allows the specification of relationships between people rather than just describing a person's contact information. It covers domain independent facts about agents and provides a common vocabulary to express relationships between people in more detail. The hierarchical structure of the T-Box constitutes the framework for describing persons, groups and organizations. Social relationships between instances can be expressed by using a variety of properties, which are themselves structured in a hierarchical order with respect to source and domain restrictions.

Complementary to the agent.owl ontology the calendar.owl ontology has been modelled to consider additional facets for ontology-based information management through the description of events. Those descriptions basically consist of the date and time when the event takes place and the people attending it. The fundamental design of the T-Box differentiates between a calendar for private events, and a second one for business purposes. Further distinctions in the structure of the ontology are made according to spatial, temporal and social circumstances of events.

3 Qualitative Reasoning

Having developed appropriate core ontologies, knowledge retrieval and information management are key areas of application that profit from knowledge sharing through seamless interoperability as well as profound reasoning support. With reference to databases of personal contacts, for instance, several other facts and assumptions can be derived by making use of ontology-based knowledge management and inference.

Thanks to the logical coherence of ontologies, specific information can be derived based on the transitivity and reflexivity of certain entities and general classification. This additional information is consequently used to complete missing data in contact databases. Figure 1 illustrates some of the social relationships of a person called Dawson Campbell. It should be noted that even if the relationship between Helen Buchanan and Dawson Campbell has not been explicitly defined, it is possible to draw the conclusion that Helen is Dawson's mother in law: Helen, in contrast to Dawson, is female (because she has been defined as Marks's wife) and has the daughter Madeleine which is herself the spouse of Michael. So, an additional and formerly unknown relationship between two entities can be established within the ontology based on the explicit represented information. The necessary reasoning steps make use of the world knowledge encoded in the ontology. The concepts relevant to the derivation described above are defined as follows.

$$\begin{aligned}
\texttt{Mother_in_law} &\equiv \texttt{Parents_in_law} \sqcap \texttt{Women} \\
\texttt{Parents_in_law} &\equiv \texttt{Person} \sqcap \exists\, \texttt{child.Spouse} \\
\texttt{Woman} &\equiv \texttt{Person} \sqcap \texttt{female} \in \texttt{gender} \\
\texttt{Spouse} &\equiv \texttt{Person} \sqcap \exists\, \texttt{spouse.Myself}
\end{aligned}$$

DL- based reasoning of the kind just sketched is rather limited. For example, it does not allow to draw the conclusion that Laurie and Michael are siblings. A concept defining siblings would need to use variables which are not part of the OWL DL. However, by using a rule language on top of OWL, such as SWRL [3], properties such as sibling can be defined easily.

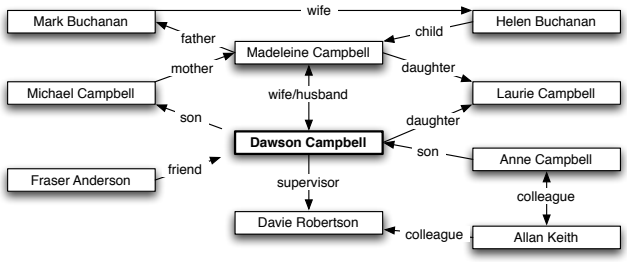


Fig. 1. Social Relations

$$(x \texttt{ child } z) \wedge (y \texttt{ child } z) \wedge (x \neq y) \Rightarrow (x \texttt{ sibling } y)$$

By feeding a rule inference engine with rules that describe properties by property chaining, such as *grandchild*, *uncle* or the *sibling* property stated above, additional facts can be derived. In doing so, implicit knowledge is again used to derive explicit information which might trigger further derivations using the standard DL-based mechanisms.
Despite rather social relationships, concepts can moreover be described using a variety of different qualitative attributes.

$$\begin{aligned}\texttt{Adult} \equiv \texttt{Grandparent} \sqcup (\texttt{Person} \sqcap (\exists\ \texttt{child}\ (\texttt{Student} \sqcup \texttt{Employee})) \sqcup \\ (\exists\ \texttt{degree}\ \texttt{Degree}) \sqcup (\exists\ \texttt{employer}\ \texttt{Person}) \sqcup (\exists\ \texttt{head}\ \texttt{Organization}))\end{aligned}$$

The information gained through the design of concepts like *Adult* allows for manifold usage in different areas of applications. To give an example, an ordinary desktop application like an email client could take advantage of the information made available in adjusting the composition of email messages with respect to the recipient. Thus, the application is able to differentiate between your 16-year old son and your business partners to adjust a variety of predefined settings.

4 Introducing McAnt

McAnt is our first prototype, which has been developed to demonstrate the use of ontology reasoning support for desktop applications. McAnt retrieves qualitative information from standard personal management applications bundled with Apple's Mac OS X, the Address Book and iCal. This information entails knowledge that can be derived through ontology-based reasoning mechanisms and in that respect serves as the basis for further application enhancements.

4.1 Components

The McAnt system consists of a number of linked components (cf. Figure 2). The McAnt application itself, which has been developed as a Java application with a native Mac OS Cocoa interface, the OWL reasoning engine Racer [4], Apple's Address Book and iCal as well as the set of core ontologies described above, providing relevant concept descriptions including the definitions of agents, spatial and temporal entities.

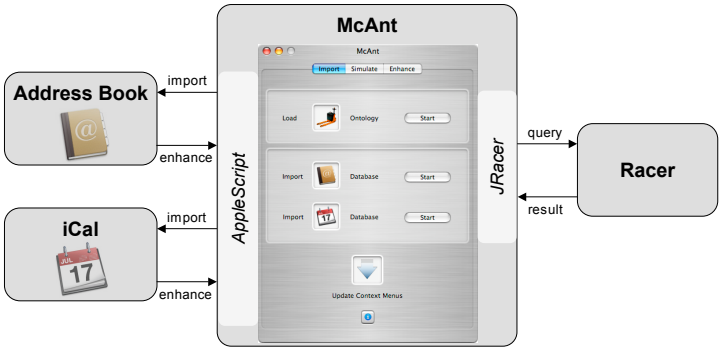


Fig. 2. McAnt Architecture

Serving as a bridge between the different components by relying on inter-application communication, McAnt accesses the Address Book database and translates it into appropriate Racer commands. The reasoning engine is accessed via JRacer, a TCP socket-based Java client for the RACER system.

4.2 Retrieving qualitative information

The retrieval of the Address Book entries has been implemented using AppleScript, a scripting language developed by Apple, which provides automation support for various complex tasks, including inter-process communication. The information stored in the Address Book comprises not only a contact database, but additionally offers the possibility to define labels describing relationships among entries including the owner.

Based on the relationship definitions, individuals that are linked with object properties are created. This enables instance reasoning support through RACER, resulting in an automatic classification within the T-Box. Thereupon the contacts have been classified and assigned to the appropriate concepts like *Family* or *Colleague*.

Furthermore, McAnt also makes use of the personal information managed with iCal. Similar to the Address Book imports, an AppleScript accesses the iCal database and translates it into appropriate RACER commands. This time the high level information retrieved consists of events, which can be associated with people attending this event and the location where it takes place. According to the concepts defined in the calendar ontology, the reasoning system classifies these event instances with respect to their properties – based on the kind of event (is it a business or private event?), its attendees as well as recurrence settings. So in case, the supervisor of the owner attends the meeting, it is therefore categorized as an important event.

4.3 Application Enhancement

Apart from organizing personal information in a taxonomic structure within the T-box, ontology-based reasoning in fact derives entailed knowledge. The entailed information can then be used to enrich applications, which in this case originally provided the data beforehand. McAnt avails this fact in the form of even “smarter” groups.

To achieve this, McAnt recursively retrieves all sub-concepts of the concept *Person* defined in the agent ontology and finally creates appropriate groups in the Address Book. Afterwards the groups are populated according to the classification results obtained from the reasoner. These "intelligent" groups help to navigate through the address book database by classifying the contacts with respect to their relationship with the owner. Unlike the rather limited possibilities of group definitions in the Address Book the ontology-based approach provides the full expressiveness of OWL DL. Latter makes use of the social relationships between the entities defined in the Address Book and therefore provides more sophisticated possibilities for creating smart folders. Furthermore, the logical consistency of the folder definitions is maintained, since conflicting definitions can be detected through the reasoner.

5 Discussion

In this work we presented our initial ideas to support Desktop Applications by rich semantic representations and reasoning mechanisms. In using extensive background knowledge formalized in ontological structures using the expressive OWL language, we were able to achieve knowledge integration across Apple's personal information management components, the Address Book and the iCal application. Furthermore, we highlighted how the functionality of those components can be enhanced by making use of ontology-based formalization and reasoning.

Taking advantage of the full strength of the OWL DL language, the smart groups mechanism implemented in Apple's Address Book application can be tremendously improved. The specification of smart folders in Apple's current implementation is limited to simple sequences of either conjunctions or disjunctions of predefined literals not covering the given social links between entries nor incorporating hierarchical groups. A further drawback of Apple's approach is the lack of any logical reasoning support. Not even simple consistency checks on the represented data are made. Therefore it easily happens that a smart group is given an unsatisfiable description.

The work reported here is very much in line with the idea of a Social Semantic Desktop [5]. A rich modeling of world knowledge (such as the framework of social relations in our agent ontology) is an essential prerequisite for standardizing, linking and wrapping information on the desktop. However, in contrast to other approaches that are based on simple RDF representations, we decided to apply the richer ontology language OWL, originally developed for the Semantic Web, and DL-based logical reasoning.

References

1. Berner-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284** (2001) 34–43
2. RDFWeb: The FOAF project (2005) <http://xmlns.com/foaf/0.1/>.
3. Horrocks, I., Patel-Schneider, P.: A proposal for an OWL rules language. In: Proceedings of the Thirteenth International World Wide Web Conference (WWW'04), ACM (2004) 723–731
4. Haarslev, V., Möller, R.: Racer system description. In: International Joint Conference on Automated Reasoning (IJCAR'01), June 18–23, 2001, Siena, Italy, Springer (2001)
5. Decker, S., Frank, M.: The Social Semantic Desktop. Technical Report DERI-TR-2004-05-02, Digital Enterprise Research Institute (DERI), Insbruck (2004)

WonderDesk – A Semantic Desktop for Resource Sharing and Management

Xiang Zhang, Wennan Shen, Yuzhong Qu

Department of Computer Science and Engineering, Southeast University,
Nanjing 210096, P. R. China
{xzhang,wnshen,yzqu}@seu.edu.cn

Abstract. We present our approach in building WonderDesk - a semantic desktop for resource sharing and management, which is a component of our WonderSpace application and designed to be an e-Science tool. We propose a desktop architecture for resource sharing and management using semantic-enriched RSS model and P2P technology. To enable fine-grained metadata for e-Science, an ontology about resource is presented as basic vocabulary to be used in constructing resource metadata along with RSS. Further, group-specific discipline ontology is proposed to enhance the semantic description of information resource within user group.

1 Introduction

Nowadays there are many ways for resource sharing. It will be beneficial to the resource consumers if resources are provided with richer metadata. Some efforts have been made on centralized resource sharing with rich metadata, such as Forum Nokia[1]. Other conspicuous efforts are made on desktop, such as Haystack [1, 2], Chandler [3] and Gnowsis [4].

In literature [5], Decker remarks on how to build a distributed desktop, and use it for online social networking. This paper presents our primary attempt – WonderDesk, as a vision that the Semantic Web and P2P technology can be used in building a semantic-enriched and distributed desktop for resource sharing and management.

WonderDesk is designed to be an e-Science tool and mostly concern about nontrivial resources that are important for research. Born as an e-Science tool, WonderDesk is strongly affected by the visionary prescribed by Vannevar Bush in his "As We May Think" [6]. Researchers often feel eager to record their fleeting inspiration for reviewing in aftertime, or exchange thoughts with neighbor researchers for a corporate brainstorm. Sometimes after reading a classical or state-of-the-art paper, the first thing a researcher want to do is to share it with colleagues immediately. The same thing happens when someone saw a presentation of a stunning technology. WonderDesk is focusing on making these activities of e-Science feasible and efficient. WonderDesk deems these objects of e-Science, e.g. paper and presentation, as resources. We design an ontology about resource as basic vocabulary

[1] http://www.forum.nokia.com/

to construct fine-grained resource metadata. We can also add metadata about topic information to an e-Science resource. Since topics of resource differ a lot, it is hard to design a comprehensive topics category. An ontology about group-specific discipline is proposed to enhance the semantic description of information resource within user group, and each topic of a resource can be point to a class in the ontology. Different groups may use different discipline ontologies, which entirely depend on their research areas.

2 WonderSpace

WonderDesk is a component of WonderSpace, which is a Semantic Web application developed in our Lab[2]. Its potential users are researcher groups.

WonderSpace is still under development, of which WonderDesk and WonderServer are two important components. WonderDesk is a desktop tool focusing on resource sharing and management, while WonderServer acts as an information integrator for all the WonderDesk peers in a group and undertake some computation extensive tasks. It affords some web services and P2P services like indexing service simultaneously. The reason we design in this way is that: when we are trying to obtain scalability, we still want to utilize some server technology to handle issues like concurrent requests and gain high performance.

The typical scenario of WonderDesk is a combined usage with a WonderServer within a research group. We named it a stand-alone WonderSpace application.

3 WonderDesk

In WonderDesk, RSS[3] model is used as the model of resource sharing, and we use RSS 1.0 specification as the syntax to describe resources metadata. This idea is from Blog (or Weblog) systems. In WonderDesk, we enrich RSS 1.0 Vocabulary to represent specific metadata of various kinds of resources.

WonderDesk is developed basing on an open source RSS Aggregator – RSSOwl[4] (The "Owl" here has no relation with the Web Ontology Language), We extended its RSS parser and equipped it with ontologies to make our vocabularies in RSS file comprehensible to the computer. More details about RSS and our ontologies will be discussed in following sections.

WonderDesk peer has the following functionalities:

- Presents an user interface to let desktop users annotate their resources;
- Shares the annotated resource with the members in a group;
- Present an enhanced query interface to help desktop user search for the shared annotated resources within a group;
- Requests and retrieves resources from other peers.

[2] http://xobjects.seu.edu.cn
[3] http://rss.userland.com
[4] http://www.rssowl.org

It is typical but not required to use WonderDesk peers combined with WonderServer. We can behold the typical scenario as an implementation of hybrid P2P architecture [7], in which WonderServer acts as a super node. Comparing to a pure P2P architecture, we think the hybrid one offers more manageability and facilities for resource query.

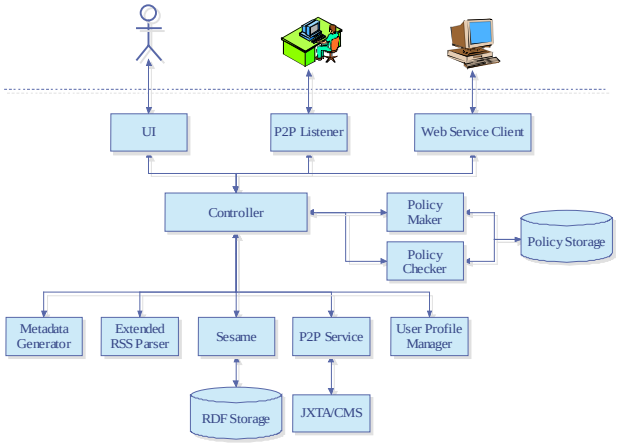


Fig. 1. WonderDesk Architecture

As presented in Fig. 1, we use Sesame[5] as the RDF framework, which has been proved to be efficient in large OWL application [8]. SeRQL[6] is a powerful RDF query language implemented in Sesame.

Some modules in the architecture are based on a java reference implementation provided by JXTA, which is a set of open protocols that allows any connected device on the network communicates and collaborates in a P2P manner [9].

4 Ontology

We have built ontology of resource, which defines terms to be used in constructing resource metadata along with other modules in RSS 1.0 specification. The other is group-specific discipline ontology describing the topic of resources.

4.1 Resource Ontology

Fig. 2 depicts part of the resource ontology model we designed. A hierarchy of "Resource" class defines different kinds of resources and their diverse properties. Although only "Information Resource " is most concerned sharable e-Science

[5] http://openrdf.org
[6] http://openrdf.org/doc/sesame/users/ch06.html

resources with WonderDesk at present, this kind of hierarchy concerning more general resources provides scalability for potential extension by any third-party.

Fig. 2. Resource Model

4.2 Group-specific Discipline Ontology

As mentioned before, topics of shared resources vary within different research groups. And in most cases, a certain group concerns about limited number of research area.

We build an ontology presenting a simple and incomprehensive classification of Computer Science research areas, where each area corresponds to a node in ACM CCS[7]. Each topic of a certain resource shared within our group can be linked to an area in this ontology.

It is nature that each stand-alone WonderSpace application will share a unified group-specific discipline ontology. We prefer the ontology to be small and specialized than huge and general for the sake of efficiency.

5 Conclusions and Future Work

We have demonstrated our approach in build WonderDesk - a semantic desktop for resource sharing and management, which is a component of our WonderSpace application. We design it to be an e-Science tool and benefit a lot from the experiences of previous research, including Haystack, Chandler and Gnowsis.

Among other semantic desktops, WonderDesk mainly contributes to the following aspects: first, we originally propose a desktop architecture for resource sharing and

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

managing using RSS model and P2P technology; second, to enable fine-grained metadata for e-Science, a resource ontology is presented as basic vocabulary to be used in constructing resource metadata along with RSS. Further, group-specific discipline ontology is proposed to enhance the semantic description of information resource within user group. These ideas have been illustrated to be feasible in our primary implementation.

Our research is just an initial attempt in building Semantic Desktop. A lot of issues are still under consideration. In the future work, we will focus on the semi-automatic or automatic annotation for resources, which will bring great facility to WonderDesk users as discussed in foregoing sections. We will also focus on the connection and collaboration of different stand-alone WonderSpace applications, and model a social researcher network. Trust and reputation issues will be addressed to help users making decision in their researching collaborations.

6 Acknowledgments

The work is supported in part by National Key Basic Research and Development Program of China under Grant 2003CB317004, and in part by the NSF of Jiangsu Province, China, under Grant BK2003001, and in part by Hwa-Ying Culture and Education Foundation. We would like to thank Dr. Zhiqiang Gao, Dr. Yuqing Zhai, Prof. Jianming Deng and Dr. Yanbing Wang for their suggestions.

Reference

1. R. Karger, D., Bakshi, K., Huynh D., Quan, D., and Sinha, V.: Haystack: A General Purpose Information Management Tool for End Users of Semistructured Data. CIDR 2005: 13-26.
2. Quan, D., Huynh, D., and R. Karger, D.: Haystack: A Platform for Authoring End User Semantic Web Applications. International Semantic Web Conference 2003: 738-753
3. The Chandler Project: http://www.osafoundation.org/
4. Sauermann, L.: The Gnowsis Semantic Desktop for Information Integration. Wissens management 2005: 39-42
5. Decker, S. and Frank, M.: The Social Semantic Desktop. DERI Technical Report. 2004.
6. Bush, V.: As We May Think. The Atlantic Monthly 176(1): 101-108 (1945)
7. Yang, B. and Garcia-Molina, H.: Comparing Hybrid Peer-to-Peer Systems. VLDB 2001: 561-570
8. Yuanbo G., Zhengxiang P. and Jeff H.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. International Semantic Web Conference 2004: 274-288
9. Groth, T.: Project JXTA. Peer-to-Peer: Ökonomische, technische und juristische Perspektiven 2002: 201-208