Semantic Desktop 2.0: The Gnowsis Experience

Leo Sauermann[1], Gunnar Aastrand Grimnes[1], Malte Kiesel[1], Christiaan Fluit[3], Heiko Maus[1], Dominik Heim[2], Danish Nadeem[4], Benjamin Horak[2], and Andreas Dengel[12]

[1] Knowledge Management Department
German Research Center for Artificial Intelligence DFKI GmbH,
Kaiserslautern, Germany
{firstname.surname}@dfki.de
[2] Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, Germany
[3] Aduna BV, Amersfoort, The Netherlands
christiaan.fluit@aduna.biz
[4] University of Osnabrueck, Germany
danzinde@gmail.com

Abstract. In this paper we present lessons learned from building a Semantic Desktop system, the gnowsis beta. On desktop computers, semantic software has to provide stable services and has to reflect the personal view of the user. Our approach to ontologies, the *Personal Information Model* PIMO allows to create tagging services like del.icio.us on the desktop. A semantic wiki allows further annotations. Continuous evaluations of the system helped to improve it. These results were created in the EPOS research project and are available in the open source projects Aperture, kaukoluwiki, and gnowsis and will be continued in the Nepomuk project. By using these components, other developers can create new desktop applications the web 2.0 way.

1 Introduction

A characteristic of human nature is to *collect*. In the information age we have moved from the basic collection of food, books and paintings to collecting websites, documents, e-mails, ideas, tasks and sometimes arguments and facts. We gather information and store them on our desktop computers, but once stored the satisfaction of possessing something is soon distorted by the task of finding information in our personal data swamp [9]. In this paper we present parts of the gnowsis semantic desktop framework, a tool for personal information management (PIM). In addition to providing an interface for managing your personal data it also provides interfaces for other applications to access this, acting as a central hub for semantic information on the desktop. The described *gnowsis* system is a prototype of a *Semantic Desktop* [17], aiming to integrate desktop applications and the data managed on desktop computers using semantic web technology. Previous work published about Semantic Desktop applications [5, 12, 15] did show that this approach is promising to support users in finding and

reminding information, and to work with information in new ways. The architecture was improved during the last years, taking inspiration from the current advanced and popularity of the *web 2.0* (See Section 3). The new architecture and new ontology policies for gnowsis version 0.9 is described in Section 2. In Section 3 we will discuss what semantic desktop applications can learn from the web 2.0, in particular we discuss how semantic wikis provide an ideal lightweight solution for ad-hoc creating of meta-data, and how such semantic wikis and tagging were integrated into gnowsis. In section 4, a short description of the information integration framework *Aperture* is given. A summary on our evaluation efforts and lessons learned is given in Section 5, indicating best practices and other remarks on practical semantic web engineering. The system has been deployed in several evaluation settings, gathering user feedback to further improve it, and an active community is now building around the gnowsis semantic desktop. As the system is now part of the EU-funded Integrated Project Nepomuk[5], which develops a comprehensive solution for a *Social Semantic Desktop*, we expect that the service oriented approach to the semantic desktop will have more impact in the future, which is the conclusion of this paper.

2 The Gnowsis Approach

Gnowsis is a semantic desktop with a strong focus on extensibility and integration. The goal of gnowsis is to enhance existing desktop applications and the desktop operating system with Semantic Web features. The primary use for such a system is *Personal Information Management* (PIM), technically realized by representing the user's data in RDF.Although the technology used is the same, communication, collaboration, and the integration with the global semantic web is not addressed by the gnowsis system. The gnowsis project was created 2003 in Leo Sauermann's diploma thesis [14] and continued in the DFKI research project EPOS[6] [7].

Gnowsis can be coarsely split into two parts, the gnowsis-server which does all the data processing, storage and interaction with native applications; and the graphical user interface (GUI) part, currently implemented as Swing GUI and some web-interfaces (See the gnowsis web-page for illustrative screenshots[7]). The interface between the server and GUI is clearly specified, making it easy to develop alternative interfaces. It is also possible to run gnowsis-server standalone, without a GUI. Gnowsis uses a *Service Oriented Architecture* (SOA), where each component defines a certain interface, after the server started the component the interface is available as XML/RPC service[8], to be used by other applications, for more detail refer to Section 3.

External applications like Microsoft Outlook or Mozilla Thunderbird are integrated via Aperture data-source (See Section 4), their data is imported and

[5] http://nepomuk.semanticdesktop.org
[6] http://www.dfki.uni-kl.de/epos
[7] http://www.gnowsis.opendfki.de/
[8] http://www.xmlrpc.com/

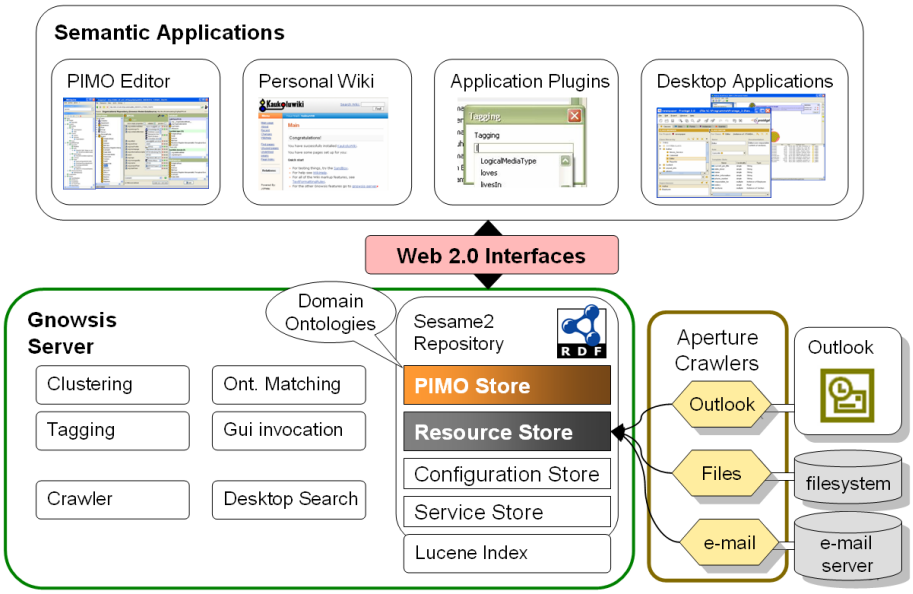

Fig. 1. The Gnowsis Architecture

mirrored in gnowsis. Some new features were also added to these applications using plugins, for example, in Thunderbird users can relate e-mails to concepts within their personal ontology (See Section 3.1).

The whole gnowsis framework is free software, published under a BSD compatible license. It is implemented in Java to be platform-independent and reuses well-known tools like Jena, Sesame, Servlets, Swing, and XML-RPC. Gnowsis can be downloaded from http://www.gnowsis.org/.

The Gnowsis Server The architecture of the *gnowsis-server* service is shown in Figure 1. Its central component is naturally an RDF storage repository. Gnowsis uses four different stores for this purpose. The PIMO store handles the information in the user's *Personal Information Model* (See Section 2.1), the resource store handles the data crawled from Aperture data-sources (See Section 4), the configuration store handles the data about available data-sources, log-levels, crawl-intervals, etc., and finally, the service store handles data created by various gnowsis modules, such as user profiling data or metadata for the crawling of data-sources.

Separating the PIMO store from the resource store was an important decision for the gnowsis architecture, and it was made for several reasons: The resource store is inherently chaotic, since it mirrors the structure of the user's applications (consider your email inbox), whereas the thoughts (eg, concepts and relations) can be structured separately in the PIMO. Another reason was efficiency, while

a user's PIMO may contain a few thousand instances for a very frequent user, it is not uncommon for people to have an archive of 100,000 emails. By separating the two we can save time and resources by only performing inference on the PIMO store. We also note that a similar approach was taken in many other projects, for instance the topic maps community, where topics and occurrences are separated [13]. A discussion of cognitive justification for such a split can be found in Section 2.1.

The storage modules in gnowsis are currently based on Sesame 2 [2] and are using Sesame's native Storage And Inference Layer (SAIL) to store the data on disk. In the previous gnowsis versions we used MySQL in combination with Jena as triple store, but this enforced users to install the database server on their desktops and also the performance of fulltext-searching in MySQL/Jena was not satisfying. By using Sesame2 with the embedded native SAIL we simplified the installation significantly. In addition to the raw RDF the PIMO and resource stores use an additional SAIL layer which utilizes Lucene[9] to index the text of RDF literals, providing extremely fast full-text search capabilities on our RDF stores. Lucene indexing operates on the level of documents. Our LuceneSail has two modes for mapping RDF to logical documents: one is used with Aperture and will index each Aperture data-object (for example files, webpages, emails, etc.) as a document. The other mode does not require Aperture. Instead one Lucene document is created for each named RDF resource in the store. The resulting Lucene Index can be accessed either explicitly through java-code, or by using special predicates when querying the RDF store. Figure 2 shows an example SPARQL query for PIMO documents containing the word "rome". The LuceneSail will rewrite this query to access the Lucene index and remove the special predicates from the triple patterns. This method for full-text querying of RDF stores is equivalent to the method used in Aduna MetaData server and Aduna AutoFocus [6].

```
SELECT ?X WHERE {
   ?X rdf:type pimo:Document ;
      lucene:matches ?Q.
   ?Q lucene:query "rome".
}
```

Fig. 2. A Query using special predicates for full-text searching.

2.1 Open Source, open standards, open minds

A key to success for a future semantic desktop is making it extendable. Partitioning the framework into services and making it possibile to add new services is one way to support this. Another contributor is exposing all programming

[9] http://lucene.apache.org/

interfaces based on the XML-RPC standard, allowing new applications can use the data and services of gnowsis.

> Open standards are important to help create interoperable and affordable solutions for everybody. They also promote competition by setting up a technical playing field that is level to all market players. This means lower costs for enterprises and, ultimately, the consumer. EU Commissioner Erkki Liikanen, World Standards Day, 14 October 2003

To create an *open standard*, others have to be able to provide a competing implementation of the standardized interfaces. The basis for this process is already laid in the commitment to HTTP interfaces and open source. The gnowsis project will be changed to a subsidiary of the bigger Nepomuk project, and all interfaces and services will be subject to a standardization and re-implementation process in the next years. By opening the project to this community, it is possible to define open standards that can be implemented by competitors, drawing new players to this market. We want to attract *open minds* to the semantic desktop, that combine existing tools creatively and can both rely on a stable platform and stable interfaces. The *web 2.0* has attracted many users. Developers recognize that by opening the data of applications and reusing existing services, new systems can be built and value can be added. We aim to transfer some of this experience to the semantic desktop creating a *semantic desktop 2.0* environment.

2.2 PIMO Ontology approach

An interesting challenge that semantic web technologies created on desktop computers is the integration of the stored data into a coherent view. In the *Personal Information Model* (PIMO) [16] approach we focus on the fact that all information is related to the user's personal view of the world. Whenever a user writes a document, reads his e-mails, or browses the web, a terminology addressing the same people, projects, places, and organizations is involved. It is connected by the interests and the tasks of the user: if the person "Paul" is working to open a new office in Rome, Italy, many documents, e-mails and files will relate to Rome and the new office. The example user Paul and parts of his PIMO are described in [16], they will be used throughout this document.

The PIMO ontology framework was initially developed in the EPOS project and consists of six components (Figure 3). The first half of these components represent mental models on a conceptual layer using formalized domain ontologies. It consists of three layers: upper-level, mid-level and domain ontologies.

PIMO-Basic, PIMO-Upper, PIMO-Mid and Domain Ontologies Apart from the native resources, the mental models are represented using a multi-layer approach. Here we transferred the 3 layer approach taken in the KnowMore-project for organizational memories (application layer, knowledge description

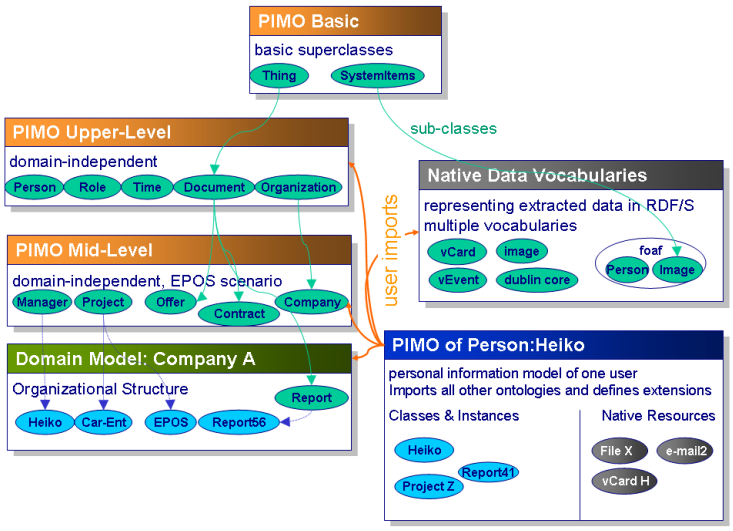

Fig. 3. PIMO ontology components

layer, information object layer; [3]) to the individual desktop. A similar approach was used by Huiyong Xiao and Isabel F. Cruz, they differentiate between *Application Layer, Domain Layer and Resource Layer* [23].

In the PIMO, the ontology layers consist of the following parts:

- PIMO-Basic: defines the basic language constructs. The class pimo-basic:Thing represents a super-class of other classes.
- PIMO-Upper: A domain-independent ontology defining abstract sub-classes of Thing. Such abstract classes are PersonConcept, OrganizationalConcept, LocationConcept, Document, etc.
- PIMO-Mid: More concrete sub-classes of upper-classes. The PIMO mid-level ontology serves to integrate various domain ontologies and provides classes for Person, Project, Company, etc.
- Domain ontologies: A set of domain ontologies where each describes a concrete domain of interest of the user. The user's company and its organizational structure may be such a domain, or it might be a shared public ontology. Domain ontologies should sub-class from PIMO-Mid and PIMO-Upper to allow integration.
- PIMO-User: the extensions of above models created by an individual for personal use.

The first three layers were created once by members of the EPOS team and are well suited for knowledge work scenarios, the domain ontologies are created for real domains and change frequently. For example, a domain ontology was

created to represent the organizational structures at the DFKI KM lab, named "*Organizational Repository*".

The PIMO of an individual user The personal mental model of the user is represented in the user's own model, called PIMO-User. Personal concepts, ideas, projects, contacts etc. are put there by the user creating classes and instances extending from the higher level PIMO classes and relations. The complete PIMO of a user is now defined as the sum of imported upper and mid-level ontologies, domain ontologies, PIMO-User, and relations to native resources found on the desktop.

Native resources and data Native resources are desktop data like existing files, e-mails, the calendar, and address book. They reflect the worker's personal view of his or her information space. A framework was created to transform several native structures to the RDF format, for example, data from Microsoft Outlook, IMAP e-mail servers, and many file formats which can be converted to RDF (See Section 4). The data extracted from these native sources is then described using a set of data-oriented RDF/S vocabularies, represented in the layer of *Native Data Vocabularies*. This data is then stored in the resource store of gnowsis.

3 Web 2.0 on the Semantic Desktop

The Web 2.0 [11] is seen as a new generation of web applications and services. Although, dismissed by many as a marketing hype and criticised for being an empty buzzword, there are several features that are generally considered to be central to the idea:

- Open data – when using some particular web-site the user should be in control of his own data, and should be able to export it and move it to an alternative service.
- Open APIs – access to site services should be possible for machine as well as for the user.
- Participatory web-sites – Known as the "architecture of participation", the idea that users create the real value of a web-site is crucial to the web 2.0. Many sites also put heavy emphasise on the social aspect of their services.
- Common features – Wikis, tagging, tag-clouds, blogs, etc. are some features are wide spread among sites considering themselves web 2.0.
- Common technology – web services (SOAP[10], XML-RPC) provide a good basis for exposing a web-site's API, AJAX (Asynchronous Javascript and XML) provides means to build fluid and elegant interfaces using HTML and RSS provides the means to stay up-to-date with a huge number of information sources without having to manually check them all.

[10] http://www.w3.org/TR/soap/

With gnowsis we are trying to take these web 2.0 features one step further and bring them "home" to the user's desktop. By having gnowsis data-sources for popular tagging web-sites, such as del.icio.us[11], Flickr[12] and BibSonomy[13], a user can import their existing tags from these sites, and integrate their personal "folksonomy" into their PIMO Ontology. This allows the user to reuse their existing classification schemes for tagging their resource, and has the added advantage of converting what used to be a flat tag list into a first-class ontology.

3.1 Tagging

Modern Tagging Systems are designed to support users in collecting new resources. Nowadays we are able to tag and manage our private pictures in Flickr, our browser bookmarks in Delicious and our scientific bibliography in BibSonomy. For these three web 2.0 services, we have developed Aperture crawlers and can integrate tag information to the resource store.

The "Ontology matcher" described in Section 4.1 allows morphing these crawled tags into personalized tags in the user's PIMO. To simplify the access to the information stored in the PIMO, we developed a tagging API which can be accessed via XML-RPC and a tagging extension for the Mozilla Thunderbird email client to use it. This enabled the user to connect incoming mails with existing tags in the user's PIMO. A similar approach in the web domain was developed by technorati[14]. The gnowsis tagging API can be seen as *technorati for an individual*.

Further development of the tagging approach in gnowsis is being done by Benjamin Horak in his diploma thesis [8]. The project, called "Contag", contains several new features for a tagging environment: It should be possible to automatically propose new instances and classes as tags for a given resource. By invoking different web 2.0 services such as those developed by Yahoo, TagTheNet[15] and Wikipedia, we can get more information about the resource at hand, and statistical analysis of these sources should allow us to propose correct tags, even if the tag is not explicitly mentioned in the resource itself.

3.2 Open data, Open APIs

All services and APIs available in gnowsis are also exposed as XML-RPC services, meaning all the gnowsis functionality can be accessed from outside the core Java part of gnowsis. The is extremely beneficial because it opens up gnowsis to a big section of other programming languages and developers. Internally we have used these interfaces to quickly develop testing interfaces for various gnowsis components using HTML, Javascript and AJAX. Using the "JavaScript

[11] http://del.icio.us
[12] http://www.flickr.com
[13] http://bibsonomy.org
[14] http://www.technorati.com
[15] http://www.tagthe.net

O Lait" library[16] calling gnowsis function from javascript is trivial. For example, consider this code-snippet from our debug interface, which uses XML-RPC to perform full-text searches on the user's PIMO:

```
<h2 class="header">query pimo</h2>
<div class="container">
q:<input type="text" id="queryFulltext" size="20">
<button onclick="gnowsis_callXmlMethod('gnowsis-server','dataaccess','querySelect',
    document.getElementById('queryFulltext').value,'fulltext');">query!</button>
</div>
```

3.3 Semantic Wiki

Traditional wikis enable people to collaboratively author a set of interlinked texts (*wiki pages*). The idea of semantic wikis is not only to edit texts but author information that can be processed by automatic means. In practice, this means that semantic wikis aim to support advanced queries ("What semantic web researchers wrote more than 60 papers?") and advanced search ("Search for *ant* as in *software*.").

Gnowsis integrates with the semantic wiki *Kaukolu*[17] [10]. The main idea is that a wiki page can be created for every instance in the PIMO-User ontology. In Paul's PIMO, there would be wiki pages for *Rome, Italy and Paul*. Note that each wiki-page is automatically a *tag*. This means that every gnowsis resource can be browsed in Kaukolu and vice versa. The same is true for relations between resources which can be created either gnowsis or Kaukolu. In gnowsis, relations are created using the standard GUI, while in Kaukolu, relations are written in a plain text syntax that is similar to N3. The user gets supported interactively with an autocompletion feature when entering data. This relieves him from having to know every relation's name or URI. The autocompletion feature bases its suggestions on ontologies stored in the PIMO-storage.The integration of Kaukolu with gnowsis opened up for several interesting features:

- **Browser integration**: With the wiki, it is possible to use the browser as a simple frontend to the gnowsis system. We even plan to move some of gnowsis' more advanced features to the wiki by way of using wiki plugins.
- **Collaborative authoring**: You can set up public spaces in the wiki to which other people are allowed to contribute. Since Kaukolu pages may also contain ontologies, this allows (simple) collaborative ontology authoring.
- **Simple data input**: Wikis are a well-known tool for authoring texts without the need to adhere to rigid templates. This can be used in the semantic desktop context, too, as with the wiki it is possible to add unstructured information (for which either no schemas exist, which are too costly to formalize, or no benefit in formalization can be thought of) to any desktop resource present in the gnowsis system.

[16] http://jsolait.net/
[17] http://kaukoluwiki.opendfki.de

4 Aperture to extract resources

To interface with applications that are not semantically enabled gnowsis uses a framework called Aperture [4]. Harvesting as much existing semantic information as possible from legacy applications benefits the user as it lowers the entry barrier to use semantic applications, e.g., when compared to approaches that rely on manual annotations. For example, information such as folder names, document authors, and creation dates can be used to provide a semantic information space with little effort required from the user, even though this information may often be shallow or imprecise.

Obtaining this information is a complex engineering task. The information is spread among a variety of source types and file formats and can often not be queried fast enough in their native format to realize the responsiveness and metadata-richness we needed for gnowsis. Crawling and indexing this information does give us these capabilities at the cost of having to keep the extracted metadata in sync with the original sources. Recently we see a tendency to incorporate such functionality in the file system [20, 22], but these approaches are still limited and operating system-specific.

The Aperture project provides various open source components for building semantic applications that harvest and index semantically rich information from various sources. Even though Aperture is still in its early stages, a growing number of applications are already making use of it. To enable this extraction Aperture offers a number of services that can be used independently or combined: *Crawlers* access an information source such as a file system, website or mail-boxes and locate all uniquely identifiable objects such as files or e-mails, identifying them by URL. Scheme-specific *DataAccessors* access these URLs and create a data structure for each of them, holding the binary data as well as any metadata provided by that scheme, e.g. file names and last modification dates. MIME type-specific *Extractors* are capable of interpreting the binary data and extracting any information supported by that file type, such as the full-text and metadata contained in a document. Finally, Aperture provides a number of utility classes for MIME type identification of binary resources, hyperlink extraction and handling of secure connections. Interfaces for crawling the contents of archives, viewing resources in their native application and storage and querying of metadata are still under development.

We have designed Aperture with the intention to provide a light-weight, expandable and adaptable framework for handling information sources. Aperture is initiated by DFKI and Aduna as a collaborative opensource project. One of the core decisions that has provided this flexibility is the use of RDF to exchange metadata between Aperture components and between Aperture and the surrounding application.

The use of RDF may have its price as programmers may find it relatively hard to handle RDF models, compared to simple Java Maps with key-value pairs. To alleviate this problem we have designed a simple Java interface called RDFContainer for handling small RDF models. RDFContainer instances are typically used to move metadata from the extractors to other components. In

gnowsis, the extracted data is stored directly (without further inference) in the resource storage RDF database. The Aperture project is independent from the semantic desktop and published at sourceforge. And it's use is encourage for anyone wishing to generate RDF datasets from external applications and sources. At the moment, Aperture supports various file types (PDF, Powerpoint, MS-Word, ...) and complex data sources like IMAP e-mail servers, Microsoft Outlook, crawling websites, flickr, del.icio.us, etc.

4.1 Ontology matcher

Once the data is represented as RDF by Aperture, it has to be aligned with the user's PIMO. As an example, the user already has an instance of class "City" which is labeled "Rome". Aperture crawled the flickr photos of the user and found a photo of an office building, tagged with "Office and Rome". The ontology matcher will now *create a relation* between the existing city "Rome" and this photo. It can also *create new instances* in the user's PIMO, if necessary. For example, the tag "Office" may be unknown, then a new instance in the ontology will be created for this tag, guessing the right class for it. When Aperture crawls larger structures, like a file system, the ontology matcher can *synchronize* parts of the user's PIMO with the file system. For example, when the users has created a list of folders for each of his projects, instances of class "Project" can be created.

In previous work published on this topic [21], term-based matching, structure-based matching, and instance-based matching algorithms were used in combination to implement such services for peer-to-peer ontology alignment and organizational memory (OM) wide ontology management. We are working to port this previous work to the current gnowsis system. In the current software package, only simple algorithms are bundled to realize the basic functionality. We want to encourage others to find more algorithms that work on this existing data.

5 Evaluation

Evaluation of Semantic Desktop that supports user with information finding and reminding with high-level PIM tasks, such as organising resources according to their mental model, needs a naturalistic flexible approach and valid and reliable methods for evaluation [1].

Gnowsis development maps closely to the spiral model of software development: our approach has been to define initial use-cases, write a prototype, and evaluate it with domain experts. A user evaluation of gnowsis based on a structured questionnaire has been made on power users to elicit their experiences, efforts have been made to enhance usability features which could be compared between different versions. We evaluated gnowsis version 0.8 in the EPOS project with domain experts [18], and this provided an important input to modify several features for the new gnowsis 0.9 architecture described in this paper.

– The possibility to add multiple tags to a document was used, in the mean 2.5 tags were attached to a file, which is significantly more than the single category a hierarchical file system provides.

– The gnowsis desktop search was used very frequently. Users answered in the questionnaire that they found unexpected information and that the categories provided by the PIMO helped during retrieval.
– The participants agreed that the PIMO reflects their personal mental models

The gnowsis 0.8 prototype had a Firefox-plugin to support link and browse from within the browser. Gnowsis version 0.9 will come include with a tagging plugin that can be seen as a replacement for today's bookmarks. The possibility to move and classify files (which we call "DropBox") as well as a semantic search are provided in both prototypes because it turned out to be a feature frequently used by the users. Only the Peer-to-Peer search was abandoned for version 0.9 (Although this will be reintroduced in the Nepomuk project). Completely new features in version 0.9 are the installation wizard that is indispensable in todays software and the PIMO. The possibility of relating resources has been completely reconsiderd. The previous linker reflected the technical nature of RDF directly in user interface, thus has been replaced by the annotator, a process-support approach on how to support linking items. Drag and drop support has also been enhanced, as it is an intuitive mechanism for invoking certain operations and its pervasive support adds flexibility in information manipulation.

5.1 Lessons learned

Building the first gnowsis prototypes has helped us to understand typical problems that arise in semantic web projects, and in this section we will enumerate our main problems. The first problem was that we had *no clear strategy regarding ontologies*. During the version 0.8 prototypes, we did allow both OWL and RDF/S semantics. Through this mixture, the inference engine of Jena had scalability problems and we disabled it. In the 0.9 version, we separated resource store from PIMO store and inference support is now only enabled for the PIMO store and there is a clear policy for the use of ontologies. The PIMO ontology approach is well documented [16] and developers can check the data model for validity, using a convenient web interface. We provided an example ontology (*Paul's Pimo*) which models the example user Paul, and because we created it before the software developments started, all developers could use it to create JUnit tests and user interface Java-Beans. Paul's Pimo accelerated development speed and improved code quality, as we had test data right from the start.

The second problem was the approach to *data extraction*. Two different approaches were evaluated [19], the first based on virtual graphs and live access to data sources. Using virtual graphs, a RDQL query to the system was translated to calls to the datasource, for example a query to list persons with certain properties was forwarded to be handled by Microsoft Outlook. This approach was complicated to implement, and the response times were unacceptable for complex queries. The second alternative was to crawl all data into a RDF database and do the queries on that database. Clearly, the second alternative had lower programming effort and better response times. In version 0.9 the database backend was also changed from Jena with MySQL storage to Sesame2 with storage to

binary files, removing the dependency of MySQL. For an end-user application, it was not acceptable to install and configure MySQL together with gnowsis.

The third problem was the attempt to *create a generic RDF interface* that allows both editing and browsing of information. Inspired by Haystack, several prototypes were build for gnowsis to edit RDF in a generic way, and the results were not accepted by end-users and abandoned. The current decision is to use *special purpose applications*; for each task a separate, specialized user interface, like the Drop-Box, linker and desktop search tools mentioned above.

Many problems were solved with approaches that resemble web 2.0 ideas — open data, open apis and a service oriented architecture.

6 Conclusion

Existing semantic desktop implementations like gnowsis version 0.8, Haystack, DBIN, or Mindraider lack the ability to integrate external applications. In theory, they are extensible but in practice programming barriers and monolithic architecture limit the extensibility. The approach taken in gnowsis 0.9 and the future Nepomuk Semantic Desktop framework aims at a service-oriented architecture. Based on interfaces to the core parts and a clear separation of components, we could improve the extensibility of the system. The use of the common database Sesame2 and its web interface allowed developers to understand the system faster and access the data through known interfaces. Also a clear guideline on managing ontologies is needed, and the PIMO approach provided this. It combines findings published by several other authors: a layered ontology, separation of native resources from ontology concepts and a personal model that can be freely edited by the user. Using web 2.0 philosophy in combination with semantic web technology, we propose the semantic desktop framework gnowsis as useful basis for future semantic desktops. An evaluation with knowledge workers and our own experience with the system has shown that the service-oriented approach supports knowledge workers.

Gnowsis will continue as part of the Integrated Project Nepomuk, where we will see more innovation on the Semantic Desktop in the next years.

Acknowledgements The PIMO ontology system was designed, and created by Ludger van Elst. We want to thank Man Luo from TU Berlin for her input in her meeting management diploma thesis. This work was supported by the German Federal Ministry of Education, Science, Research and Technology (bmb+f), (Grant 01 IW C01, Project EPOS: Evolving Personal to Organizational Memories) and by the European Union IST fund (Grant FP6-027705, project Nepomuk).

References

1. *Evaluating Personal Information Management Behaviors and Tools.* Vol.49.No.1,Communications of ACM, 2006.

2. Seseame 2. http://www.openrdf.org/, 2006.
3. Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann, Otto Kühn, and Michael Sintek. Toward a Technology for Organizational Memories. *IEEE Intelligent Systems*, June 1998.
4. Aperture. http://aperture.sourceforge.net, 2005.
5. Adam Cheyer, Jack Park, and Richard Giuli. Iris: Integrate. relate. infer. share. In *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland*, 2005.
6. Christiaan Fluit. Autofocus: Semantic search for the desktop. In *9th Int. Conf. on Information Visualisation, London, UK*, pages 480–487, 2005.
7. H. Holz, H. Maus, A. Bernardi, and O. Rostanin. From Lightweight, Proactive Information Delivery to Business Process-Oriented Knowledge Management. *Journal of Universal Knowledge Management*, 0(2):101–127, 2005.
8. Benajmin Horak. Contag : Atagging system linking the semantic desktop with web 2.0. Master's thesis, University of Kaiserslautern, 2006.
9. Aura Lippincott Jason Frand. Personal knowledge management: A strategy for controlling information overload. 2002. draft.
10. Malte Kiesel. Kaukolu – hub of the semantic corporate intranet. In *Proc. of Semantic Wiki Workshop at the ESWC 2006*, 2006.
11. Tim O'Reilly. What is web 2.0, design patterns and business models for the next generation of software.
12. 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.
13. Holger Rath. The Topic Maps Handbook. empolis white paper, empolis GmbH, 2003.
14. Leo Sauermann. The gnowsis-using semantic web technologies to build a semantic desktop. Diploma thesis, Technical University of Vienna, 2003.
15. Leo Sauermann. The semantic desktop - a basis for personal knowledge management. In *Proceedings of the I-KNOW 2005.*, pages 294 – 301, 2005.
16. Leo Sauermann. Pimo-a pim ontology for the semantic desktop (draft). Draft, DFKI, 2006.
17. Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and outlook on the semantic desktop. In *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*.
18. Leo Sauermann, Andreas Dengel, Ludger van Elst, Andreas Lauer, Heiko Maus, and Sven Schwarz. Personalization in the EPOS project. In *Proc. of the Semantic Web Personalization Workshop at the ESWC Conference*, 2006.
19. Leo Sauermann and Sven Schwarz. Gnowsis adapter framework: Treating structured data sources as virtual rdf graphs. In *Proceedings of the ISWC 2005*, 2005.
20. Apple Spotlight. http://www.apple.com/macosx/features/spotlight/, 2004.
21. Ludger van Elst and Malte Kiesel. Generating and integrating evidence for ontology mappings. In *Proc. of the 14th EKAW*, volume 3257 of *LNAI*, pages 15–29, Heidelberg, 2004. Springer.
22. Microsoft WinFS. http://msdn.microsoft.com/data/winfs/default.aspx, 2006.
23. Huiyong Xiao and Isabel F. Cruz. A multi-ontology approach for personal information management. In *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland*, 2005.