The NEPOMUK Project - On the Way to the Social Semantic Desktop

Tudor Groza, Siegfried Handschuh, Knud Möller
(DERI, National University of Ireland, Galway, Ireland
tudor.groza@deri.org, siegfried.handschuh@deri.org, knud.moeller@deri.org)

Gunnar Grimnes, Leo Sauermann
(DFKI Kaiserslautern, Germany
gunnar.grimnes@dfki.de, leo.sauermann@dfki.de)

Enrico Minack
(L3S Hannover, Germany, minack@l3s.de)

Mehdi Jazayeri, Cédric Mesnage
(Faculty of Informatics, University of Lugano, Switzerland
mehdi.jazayeri@unisi.ch, cedric.mesnage@lu.unisi.ch)

Gerald Reif
(University of Zurich, Switzerland, reif@ifi.unizh.ch)

Rósa Gudjónsdóttir
(Royal Institute of Technology, KTH, Sweden, rosag@kth.se)

Abstract: This paper introduces the NEPOMUK project which aims to create a standard and reference implementation for the Social Semantic Desktop. We outline the requirements and functionalities that were identified for a useful Semantic Desktop system and present an architecture that fulfills these requirements which was acquired by incremental refinement of the architecture of existing Semantic Desktop prototypes. The NEPOMUK project is primarily motivated by three real-life industrial use-cases, we briefly outline these and the processes used to extract required functionalities from the people working in these areas today, and we present a selection of typical tasks where the Semantic Desktop could be of benefit.
Key Words: Semantic Desktop, Personal Information Management, Semantic Middleware
Category: H.3.7, H.5.4

1 Introduction

In traditional desktop architectures, applications are isolated islands of data – each application has its own data, unaware of related and relevant data in other applications. Individual vendors may decide to allow their applications to interoperate, so that e.g. the email client knows about the address book. However, today there is no consistent approach for allowing interoperation and a system-wide exchange of data between applications. In a similar way, the desktops of

different users are also isolated islands - there is no standardized architecture for interoperation and data exchange between desktops. Users may exchange data by sending emails or upload it to a server, but so far there is no way of seamless communication from an application used by one person on their desktop to an application used by another person on another desktop.

The problem on the desktop is similar to that on the Web. On the Web we are faced with isolated data islands, and also as on the desktop there is not yet a standardized approach for finding and interacting between applications.

The Social Semantic Desktop (SSD) paradigm adopts the ideas of the Semantic Web, which offers a solution for the Web. Formal ontologies capture both a shared conceptualization of desktop data and personal mental models. RDF serves as common data representation. Web Services - applications on the Web - describe their capabilities and interfaces in a standardized way and thus become Semantic Web Services. On the desktop, applications (or rather: their interfaces) will therefore be modeled in a similar fashion. Together, these technologies provide a means to build the semantic bridges necessary for data exchange and application integration. The SSD will transform the conventional desktop into a seamless, networked working environment, by loosening the borders between individual applications and the physical workspace of different users.

The aim of the NEPOMUK project[1], described in this paper, is to provide a standardized description of a SSD architecture, independent of any particular operating system or programming language. Reference implementations will show the feasibility of the standard. The paper is structured as follows: we start with Section 2 by describing the engineering cycle we follow in the project. Then we detail in Section 3 scenarios captured from real-world case-studies and in Section 4 a list of functionalities extracted from these scenarios. Section 5 depicts the current version of the NEPOMUK SSD Architecture, while Section 6 shows related approaches for building the SSD. In Section 7 we state our conclusions.

2 NEPOMUK **Engineering Cycle**

The NEPOMUK project relies heavily on existing software developed by the partners. On the other hand, usability research is being held with the case studies partners by interviewing potential users of the SSD. This specific set up of the project led us to develop our engineering cycle (Figure 1). This cycle represents the way we intend to merge the existing technologies and the needs from users.

Clockwise, Figure 1 shows the forward engineering cycle. We analyzed the end-user's intended *usage* of the SSD, studied the different use cases and formulated them into *scenarios*. We generalized the individual scenarios and extracted the common *functionalities* that make up the SSD. These functionalities formed

[1] http://nepomuk.semanticdesktop.org/

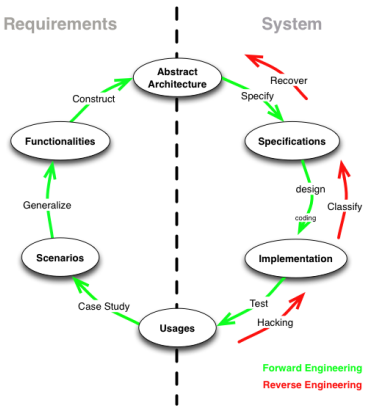

Figure 1: NEPOMUK Engineering Cycle

the basis to define the reference *architecture* which in turn lead to the service *specification* and *implementation* that is tested by the end-users. On the other hand, partners already started to hack components that are likely to be needed by the SSD or had component developed before the NEPOMUK project started. Therefore, we reverse engineered these components to get their specifications and used the gained experience when defining the architecture.

The construction of the architecture of the SSD is therefore the combination of different parts, (1) the requirements and objectives from the vision of the SSD driving the NEPOMUK project, (2) the functionalities from user studies (forward engineering), and (3) the service specifications of existing implementations (reverse engineering). The overlaps between these areas give us confidence in the needs. Combining these three sources results in a complete architecture. Thus the architecture represents a shared understanding of all partners involved in the project and we see it as a roadmap towards the realization of the SSD.

In the next sections we present some of the scenarios which we considered as being particularly representative for the SSD paradigm, then we show the list of functionalities abstracted from the user study material.

3 Scenarios

The study of user needs regarding collaboration on the SSD is a major goal of the NEPOMUK project. User studies were carried out in the project at the case study partner sites, which are companies and research labs working in the area of business software, biomedical research, Linux development, and management consulting. The type of work performed varies between the case study partners, but what they have in common is the fact that the employees are knowledge

workers, receiving, interpreting and structuring information on a daily basis. The purpose of the user studies was to understand the work environment in order to develop a SSD that meets the knowledge workers' needs and requirements. 40 contextual interviews [4] and seven video brainstorming workshops [9] were performed with employees at the different partner sites. To document the resulting user requirements 14 personas and 40 usage scenarios [6, 7] were created, illustrating the user needs, desires and expectations on the SSD. Personas are fictitious persons that represent different user groups and are always based data collected in user studies. A persona and a related scenario where the persona uses the SSD that we will develop, is an effective way to illustrate how the users want the SSD to operate. In this section we summarize a collection of the usage scenarios with the help of our primary personas.

Dirk gets task from Claudia. Claudia is working on a project deliverable and she identifies tasks to be done. She adds the tasks to the project and assigns them to Dirk. Dirk is notified of his new tasks and he accepts the responsibility for some of the tasks and Claudia is notified. Dirk realises that some tasks require more specific knowledge so he declines them and suggests allocating them to Martin. Claudia reassigns the tasks to Martin.

Josephine follows-up the project plan. Josephine is following up on an active project she is administrating. It involves Karen and a few other trainers. She accesses the project plan and browses to see if everything is on schedule. The project plan is connected to the trainers calendars and all changes in their calendar get fed directly into the project plan.

Karen edits a document with another person. Karen is to give a presentation for an existing client in a couple of days. The presentation is new and the purpose is to sum up a series of training programs she has performed for the client. When she is working on a slide she can see a new graphical layout suggestion by Josephine who is concurrently working on the presentations graphic form. The system allows them to collaborate, make changes, discuss and explain their intentions and thoughts.

Karen shares experience. Karen finds the time to take care of some administration issues. She just finished a project successfully and feels that the experiences should be shared with her colleagues. She opens the course material, marks it Shared, and adds a few keywords to make sure that people interested find the material.

4 Functionalities

In order to integrate the requirements expressed in the scenarios and other materials produced in the case studies we need to use a more formal approach. All the material was processed by a group of members of the project coming

from different areas: developers, case study partners, architects and usability designers. The results of this workshop is an homogeneous list of functionalities required to satisfy the scenarios. For each functionality, we provide a name, a short textual description, inputs, outputs and the relevant material in which this functionality was discovered. We grouped these into five cluster:

Search enables users to *search* for resources amongst different sources (either locally or on the network). Users also need to find relevant resources by querying by example.

Desktop. On their desktop, users *manage resources*, they use legacy applications to either create or edit documents therefore NEPOMUK needs to *integrate these applications*. NEPOMUK should provide a *notification management* system for the user to receive informations regarding shared resources and configure the ways she is notified. Even when *offline*, users should be able to access relevant resources transparently. We see *desktop sharing* as the ability to share applications or windows.

Profiling by *logging* the user's activity, NEPOMUK should be *trained* to behave according to the specific user's needs. This automatic behaviours must be *tailorable* and include *annotations* and information regarding *trust* with other users or sources(*i.e.,* if a user do not trust an information source, he should not receive results from this source).

Data Analysis. To ease semantic annotation of unstructured documents, such as text, users can use *keyword extraction*. Search results might need to be rearranged using *sorting and grouping*. The use of *reasoning* provides with new information.

Social. At the social level, the *management of groups and users* enhances *social interaction* and ease *resource sharing*. *Access rights management* tackles with the security needs. Users can *publish and subscribe* to relevant stream of information, such as the modifications made to a particular resource or the results of a search.

The discussion around these functionalities lead to the architecture which integrates the user requirements and the SSD vision. This architecture is discussed in the following section.

5 Architecture

In this section we present an overview of the NEPOMUK architecture. The architecture, as show in Figure 2, is organized in three layers. The NEPOMUK SSD is made up by the user's individual desktops which are organized in a peer-to-peer (P2P) fashion. To support the communication between the peers, the lowest layer is the *Network Communication* layer. This layer provides an *Event-based System*, which is responsible for the distribution of the events on between

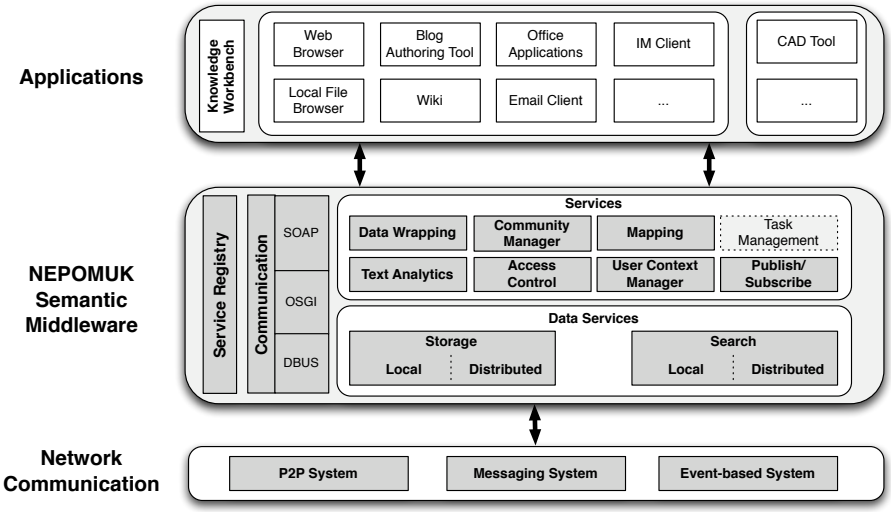

Figure 2: Layered NEPOMUK Architecture.

the NEPOMUK peers. The events carry an RDF graph as payload describing the cause of the event. The *Messaging System* routes the messages to receiver. The *Peer-to-Peer File Sharing System* enables the shared information space. It will be based on GridVine [2]. GridVine is based on P-Grid [1] and provides a distributed index which supports RDQL search queries.

On top of the Network Communication Layer, the NEPOMUK Semantic Middleware provides the core services the NEPOMUK SSD is made up from. The goal of the NEPOMUK project is to propose a reference architecture for the SSD that can be implemented on top of different operating systems such as MS Windows, MacOS, and Linux. Hence, different communication techniques such as SOAP over HTTP, OSGI[2], or D-Bus[3] can be used for interaction between the NEPOMUK services depending on the platform. Therefore, we decided to use WSDL as communication technique and programming language independent interface definition language to specify the service interfaces. The services have to register at the *Service Registry*.

The *Data Services* are responsible to control the insertion, modification, and deletion of resources on the NEPOMUK desktop. A resource can be a user, a document, a calendar entry, an email, and so on. It provides a service to store the RDF meta-data in the *Local Storage*. A resource and their RDF description

[2] OSGi Alliance – `http://www.osgi.org/`
[3] D-Bus – `http://www.freedesktop.org/wiki/Software/dbus`

can either be manually added to the NEPOMUK desktop or the *Data Wrapper* or the *Text Analysis* service extracts the information form desktop applications such as email clients of calendar applications. The Data Wrapper will be used to extract meta-data form structured data sources (*e.g.,* email headers, calendar entries, *etc.*) and will be implemented based on Aperture [3]. The Text Analysis service will be used to extract meta-data form unformatted text (*e.g.,* email bodies, text processor documents, *etc.*). For local queries and for offline working the RDF meta-data is stored in the Local Storage. If a resource is shared with other users in an information space, the meta-data is also uploaded to the distributed index of the P2P file sharing system. The *Search* service can either issue a *local* search in the local storage or a *distributed* search in the underlying P2P system.

Ideally only one ontology exists for a domain of interest such as contact data, calendar events. In reality, however, we are faced with many ontologies of (partly) overlapping domains (*e.g.,* foaf and vCard for contact data). Therefore, the NEPOMUK middleware provides a *Mapping Service* that can be used by other middleware services and services in higher layers to translate RDF graphs from a source ontology to a target ontology.

Actions a user performs on the shared information space have to be approved by the *Access Control* System. Depending on the group membership of a user, maintained in the User/Group Management, the *Community Management* grands the privileges to perform the action. The access rights, the user, and the group data are stored as RDF graphs in the distributed index of the peer-to-peer system. This data is encoded using the access right ontology and the user/group ontology, which belong to the NEPOMUK core ontologies.

The NEPOMUK middleware logs the actions a user performs on the resources on his desktop. The logged data is stored in the Local Storage and is analyzed by the *User Context Manager* to capture the current working context of the user. The working context of the user is used to suggest meaningful actions to the user depending on the task a user is currently working on.

The *Publish/Subscribe System* allows users to subscribe to events in the NEPOMUK system. The subscriptions are stored as SPARQL queries [10] which are matched against the RDF payload of the events. When the subscription, *i.e.,* the SPARQL query, matches the event, the *Messaging System* looks up the preferred notification media (*e.g.,* email, instant messaging, SMS) and delivers the messages. The Messaging System is further used for synchronous and asynchronous communication between NEPOMUK users.

The NEPOMUK Middleware provides the core services of the NEPOMUK architecture. These services can be accessed via the NEPOMUK API. An application programmer can build usage specific services on top of the NEPOMUK API. By using the functionality provided by the API, the programmer can implement new functionality according to the end-users' business requirements.

Hence, the basic set of services provided by the NEPOMUK API can be customized and extended by businesses and organizations. For example, a company might be interested in integrating *Task Management* system whereas another might be interested in having document versioning support for resources. The end-user specific services are shown in dashed boxes in Figure 2.

The top layer of the architecture is the presentation layer. It provides a user interface to the services provided by the NEPOMUK desktop. The presentation layer is built using the NEPOMUK API. Many desktop applications are possible sources for resources that should be managed by NEPOMUK. Therefore, each desktop application should integrate support for the NEPOMUK Middleware. Since this assumption does not hold for most applications, we developed plug-ins and add-ons to enable a seamless integration for popular applications such as the MS Office Suite, which for example extract email or calendar data and adds them as resources to the NEPOMUK desktop. However, with in the NEPOMUK project we develop applications such as *Wikis* or *Blog Tools* that have generic support for the SSD and build directly on tho of the NEPOMUK API.

In addition, the *Knowledge Workbench* is the central place to browse, query, view, and edit resources and their meta-data. This way the Knowledge Workbench aims to replace current file management tools such as the File Explorer. If the SSD is extended by usage specific services, the application programmer has also to provide the corresponding user interface in the Presentation Layer.

6 Related Work

In the following we review the most important projects related to establishing a SSD. These projects are coming from the research, business, as well as the open-source community. After a brief general overview of each project, we want to learn from the related work as the conclusion of this section.

Gnowsis Semanic Desktop. The first research project targeting a Semantic Desktop system is the *Gnowsis Semantic Desktop* [12]. Its goal is to complement established desktop applications and the desktop operating system with Semantic Web features, rather than replacing them, while primarily focusing on *Personal Information Management* (PIM). The thesis addresses the problems of how to identify and represent desktop resources in an unified RDF graph.

Haystack. A major research project concerning an integrated approach in our field is *Haystack* [11]. Application-created barriers of information representation and accessibility are removed by simply replacing these applications with Haystack's word-processors, email client, image manipulation, instant messaging and other functionality. Haystack was ground-breaking in terms of the dynamic creation of user interfaces, but ended before establishing any standards.

Semex. Another relevant Personal Information Management tool is *Semex* (SEMantic EXplorer) [8]. Semex concentrates on the problem of Reference Rec-

onciliation, meshing objects and relations seamlessly together. They combine three measures for this approach being evaluated on one of the author's private dataset. In contrast, NEPOMUK will add more reconciliation algorithms from the Semantic Web and evaluate the data integration in industry scenarios.

IRIS. The idea of the PIM system *IRIS* [5] is to have an integrated environment, similar to Haystack, but based on standard software, which is integrated into one coherent interface, allowing to classify and display related information. By today, the project lists only one publication introducing their approach.

Apogée. The *Apogée* project aims at building a framework to create Enterprise Development Process-oriented desktop applications, independent from vendor or technologies. Probably due to its status of an industrial project, it aims at implementing state-of-the-art features, but not beyond.

All integrated Semantic Desktop systems faced similar problems. First problem is evaluation and verification of the ideas in industry settings. Most systems like IRIS or Semex are not evaluated yet, they are only used by the developers in self-experiments. With its case studies, NEPOMUK will provide a testbed to show the implications of the whole Semantic Desktop in both, industrial environments and open-source communities.

Second problem is that the projects do not consider collaborative work and the interconnection of Semantic Desktops at all. They concentrate on a single user scenario, whereas NEPOMUK also tackles collaborative knowledge work.

Last and probably most significant problem is integration. While for example DBin shows the aspect of collaborative work, it does not connect to desktop applications. Though Haystack provides a well evaluated user interface, it does not re-use established Desktop applications users are used to, thus faces the user with a new environment. Further, none of the projects established standards which would increase interoperability and reusability.

Each system accommodates singular beneficial features, but also suffers from flaws like usability problems, bad performance, or missing functionality. These projects are designed as an integrated system, and despite the fact that most prototypes are open-source, it is not straightforward to reuse components of one for the other in order to amplify their features and extinguish their weaknesses. In contrast, NEPOMUK will establish a framework and standards so that components can be reused and are interoperable, creating a better whole.

7 Conclusion

This paper has given a very brief overview of the motivations, goals and progress of the NEPOMUK project. We have described the features and functionalities that our vision of a Social Semantic Desktop requires, based on observation of real knowledge-workers and their struggle with information integration using

today's technology. Using an engineering process where we worked backwards from the desired functionalities and requirements, while simultanously refined a collection of existing Semantic Desktop research prototypes, we devised an architecture for the Semantic Desktop. This architecture enabled us to build a prototype featuring some of the required functionalities, and it is released as open-source and is available for download from the NEPOMUK Web-site[4].

The core aim of the NEPOMUK project is to specify an standard for Semantic Desktop communication and processing. We are basing our work on well-established standards of Web and Semantic Web technologies, and we hope that our Semantic Desktop standards in turn will provide a fertile ground for future projects. By having a flexible and easily extendible architecture we hope that over the next years any developer looking to solve information integration problems on the desktop will look to NEPOMUK as a framework for their projects, thus by the time the project is ending, NEPOMUK will have become a useful entity in it's own right, with an active community and untold possibilities.

References

1. K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-grid: a self-organizing structured p2p system. *SIGMOD Record*, 32(3):29–33, 2003.
2. K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. In *3th International Semantic Web Conference ISWC 2004*, pages 107–121. Springer Verlag, 2004.
3. Aperture a java framework for getting data and metadata, Last visited March 2007. `http://aperture.sourceforge.net/`.
4. H. Beyer and K. Holtzblatt. *Contextual Design ? Defining Customer-Centered Systems*. Academic Press, San Diego.
5. A. Cheyer, J. Park, and R. Giuli. Iris: Integrate. relate. infer. share. In S. Decker, J. Park, D. Quan, and L. Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.
6. A. Cooper. *The Inmates are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*. SAMS, Indianapolis, 1999.
7. A. Cooper and R. Reinman. *About Face 2.0: The Essentials of Interaction Design*. John Wiley & Sons, 2003.
8. X. Dong and A. Y. Halevy. A platform for personal information management and integration. In *CIDR*, pages 119–130, 2005.
9. E. Mackay, A. Ratzer, and P. Janecek. Video artifacts for design: bridging the gap between abstraction and detail. In *Designing interactive systems: processes, practices, methods, and techniques, DIS '00*. ACM Pres, 2000.
10. E. Prud'hommeaux and A. S. eds. SPARQL query language for RDF. W3C Working Draft, 4. Octoberl 2006. `http://www.w3.org/TR/rdf-sparql-query/`.
11. D. Quan, D. Huynh, and D. R. Karger. Haystack: A platform for authoring end user semantic web applications. In *International Semantic Web Conference*, pages 738–753, 2003.
12. L. Sauermann. The gnowsis-using semantic web technologies to build a semantic desktop. Diploma thesis, Technical University of Vienna, 2003.

[4] `http://dev.nepomuk.semanticdesktop.org`