

The Social Semantic Desktop

A New Paradigm Towards Deploying the Semantic Web on the Desktop

Ansgar Bernardi², Stefan Decker¹, Ludger van Elst², Gunnar Aastrand Grimnes²,
Tudor Groza¹, Siegfried Handschuh¹, Mehdi Jazayeri³, Cédric Mesnage³,
Knud Möller¹, Gerald Reif⁴, Michael Sintek²

¹DERI, National University of Ireland, Galway,
IDA Business Park, Lower Dangan, Galway, Ireland
{Stefan.Decker, Tudor.Groza, Siegfried.Handschuh, Knud.Moeller}@deri.org

²German Research Center for Artificial Intelligence (DFKI) GmbH, Kaiserslautern,
Trippstadter St. 122, 67663, Kaiserslautern, Germany
{Ansgar.Bernardi, Ludger.Van_elst, Gunnar.Grimnes, Michael.Sintek}@dfki.de

³Faculty of Informatics, University of Lugano,
Via Giuseppe Buffi 6, 6900, Lugano, Switzerland
mehdi.jazayeri@unisi.ch, cedric.mesnage@lu.unisi.ch

⁴Software Evolution and Architecture Lab, University of Zürich,
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
reif@ifi.uzh.ch

This chapter introduces the general vision of the Social Semantic Desktop (SSD) and details it in the context of the NEPOMUK project. It outlines the typical SSD requirements and functionalities that were identified from real world scenarios. In addition, it provides the design of the standard SSD architecture together with the ontology pyramid developed to support it. Finally, the chapter gives an overview of some of the technical challenges that arise from the actual development process of the SSD.

Keywords: Data Extraction, Data Mining, Data Modeling, Information Network, Knowledge Classification, Knowledge Discovery, Meta Model, Metadata, Middleware, Semantic matching, Knowledge Management, Knowledge Creation, Knowledge Acquisition, Knowledge Sharing, Knowledge Storage,

The Social Semantic Desktop

A New Paradigm Towards Deploying the Semantic Web on the Desktop

This chapter introduces the general vision of the Social Semantic Desktop (SSD) and details it in the context of the NEPOMUK project. It outlines the typical SSD requirements and functionalities that were identified from real world scenarios. In addition, it provides the design of the standard SSD architecture together with the ontology pyramid developed to support it. Finally, the chapter gives an overview of some of the technical challenges that arise from the actual development process of the SSD.

INTRODUCTION

A large share of everybody's daily activities centres around the handling of information in one way or the other. Looking for information, digesting it, writing down new ideas, and sharing the results with other people are key activities both in work as well as in manifold leisure activities. The abundance of PCs and the Web in today's world result in new numbers and qualities of information exchange and interaction which are seen both as chance and as threat by the users. Supporting personal and shared information handling is thus a highly requested but yet unsolved challenge.

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 uploading 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 – also there, we are faced with isolated islands of data and no generic way to integrate and communicate between various Web applications (i.e., Web Services). The vision of the SW offers solutions for both problems. RDF¹ is the common data format which builds bridges between the islands, and Semantic Web Service technology offers the means to integrate applications on the Web.

The Social Semantic Desktop (SSD) paradigm adopts the ideas of the SW paradigm for the desktop. Formal ontologies capture both a shared conceptualization of desktop data and personal mental models. RDF serves as a common data representation format. Web Services – applications on the Web – can describe their capabilities and interfaces in a standardized way and thus become Semantic Web Services. On the desktop, applications

¹ RDF: <http://www.w3.org/RDF/>

(or rather: their interfaces) will therefore be modelled in a similar fashion. Together, these technologies provide a means to build the semantic bridges necessary for data exchange and application integration. The Social Semantic Desktop 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.

By realizing the Social Semantic Desktop, we contribute to several facets of an effective personal information handling:

- We offer the individual user a systematic way to structure information elements within the personal desktop. Using standard technology to describe and store structures and relations, users may easily reflect and express whatever is important in their personal realm.
- Standardized interfaces enable the integration of all kinds of available desktop applications into the personal information network. Investments in programs, data collections, and hard-learned working styles are not lost but augmented and connected into a comprehensive information space.
- Based on the SW technology basis, all kinds of automated and semi-automatic support are possible, like, e.g., text classification services, image categorization, document relevance assessments, etc.
- The exchange of standard data formats between individual work spaces is supported not only on the technical level (e.g., standard communication protocols), but also on the semantic level (via sharing and alignment of ontologies and the corresponding annotated information elements). The integration with formal ontologies eases the sharing and understanding between different persons.
- Ultimately, we thus contribute to a solution for the initialization problem of the SW: As the individual user will receive immediate benefit from the semantic annotation within the personal workspace, the motivation is high to invest the necessary structuring and formalization work. As the standards used allow for an effortless sharing of such work, the amount of semantically annotated information which can be made available in the Web grows dramatically – which in turn makes it worthwhile to develop new SW-based services.

In this chapter we describe in detail the core components which are necessary for building a Social Semantic Desktop. We illustrate the necessary standard framework and describe the role and structure of the ontologies which support the spectrum from personal to social information handling. We outline the implementation decisions which need to be observed in order to realize a consequently ontology-oriented system, which is able to deal with the numerous flexibilities required within the Semantic Web. Finally, we show examples of the benefits obtained from the realization and use of an SSD.

The ideas and implementation principles presented in this chapter are distilled from our experiences in the NEPOMUK Project². For each section we will describe the general motivation and principles and then give details on how the particular challenges have been solved in the NEPOMUK project.

² The NEPOMUK Project is supported by the European Union IST fund, grant FP6-027705

BACKGROUND

The Social Semantic Desktop vision has been around for a long time: visionaries like Vannevar Bush (1945) and Doug Engelbart (1962) have formulated and partially realized these ideas. However, for the largest part their ideas remained a vision for far too long, since the necessary foundational technologies were not yet invented – figuratively speaking, these ideas were proposing jet planes when the rest of the world had just invented the parts to build a bicycle. Only in the recent years several technologies and research streams began to provide a foundation which will be combined and extended to realize the envisioned collaborative infrastructure of the SSD.

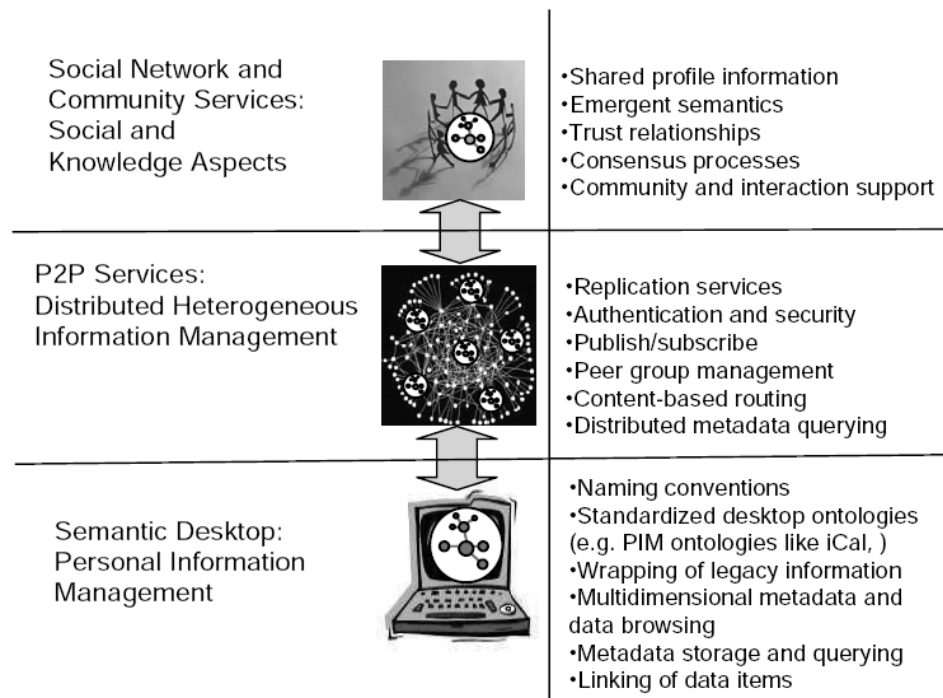


Fig. 1. Component Architecture of the Social Semantic Desktop

Fig. 1 shows the highest-level architecture and connections between components of the SSD, i.e., the social networks, the P2P infrastructure, and the individual desktops. Traditional semantics, knowledge representation, and reasoning research are now interacting. While none of them can solve the problem alone, together they may have the explosive impact of the original Web:

The Semantic Web effort provides standards and technologies for the definition and exchange of metadata and ontologies. Available standard proposals provide ways to define the syntax (RDF) and semantics of metadata based on ontologies (Web Ontology Language – OWL (McGuinness et. al, 2004), RDF Schema – RDFS). Research covering data transfer, privacy, and security issues is now also under development.

Social Software maps the social connections between different people into the technical infrastructure. As an example, Online Social Networking makes the relationships between individuals explicit and allows the discovery of previously unknown relationships. The most recent Social Networking Sites also help form new virtual communities around topics of interest and provide means to change and evolve these communities.

P2P and Grid computing develops technology to network large communities without centralized infrastructures for data and computation sharing. P2P networks have technical benefits in terms of scalability and fault tolerance, but a main advantage compared to central sites is a political one: they allow to build communities without centralized nodes of control, much as the Internet grew as fast as it did because it was based on reciprocity – it avoided political debate as to who gets to own big, expensive central facilities. Recent research has provided initial ways of querying, exchanging and replicating data in P2P networks in a scalable way.

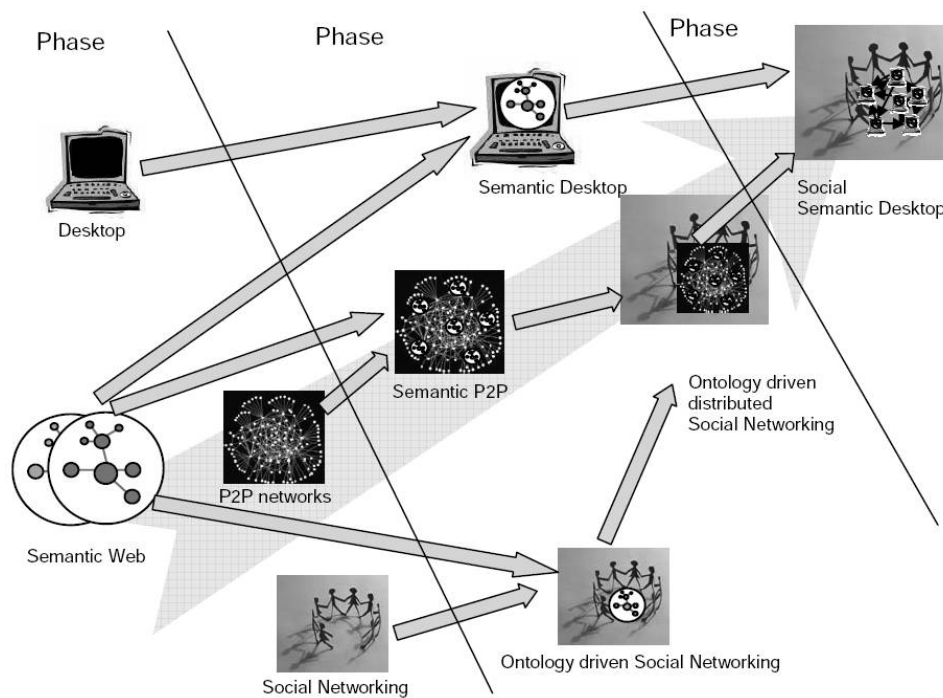


Fig. 2. Phases Towards the Social Semantic Desktop

By projecting the trajectory of current trends, we can simplify this picture by stating that next generation desktop applications will support collaboration and information exchange in a P2P network, connecting online decentralized social networks, and enabling shared metadata creation and evolution by a consensus process. The result of this process will be the Social Semantic Desktop. Fig. 2 depicts the phases in which the relevant co-evolving technologies are combined to achieve the final goal, i.e., the realization of the Social Semantic Desktop.

SCENARIOS

Before we move on to the specific functionalities that a Social Semantic Desktop supports and discuss how they are implemented, we will first present some scenarios that will illustrate what an SSD is, how it will be used, and how it will change the way we do knowledge work. We chose the scenarios such that they illustrate the different dimensions of an SSD: Sect. 3.1 describes example usage that shows the use of semantics on the desktop, and Sect. 3.2 will show the social dimension of an SSD, i.e., the interaction between desktops of different users. The scenarios give an overview of what is possible and how the SSD presents itself to the user.

The Semantic Dimension

A typical use of a single Semantic Desktop is to organize ones data: files, emails, pictures, etc. Users are able to tag those information resources with concepts from a network of ontologies. The ontologies also contain relations (or properties, to use RDF terminology), which can be used to link information resources on the desktop. Organizing information resources in this way helps users to find what they are looking for quicker, and makes it possible for the Semantic Desktop itself to aid the user in their daily work. When a user first begins using the Semantic Desktop, many often-used concepts and properties are already present. E.g., there are basic concepts such as **Person**, **Meeting** or **Place**, and properties such as **knows** or **located-in**. Also, we can assume that useful things like an ontology of all countries are already in place. Then, as the need arises, users can extend the existing ontologies – e.g., they can add concepts for a particular meeting they attend or people they know, such as **Executive-Committee-Meeting-07/06/07**, **Jane-Doe** or **Hans-Schmidt**. The following two scenarios give examples of this kind of Semantic Desktop usage. We will use two imaginary users (*personas*³) flesh them out: Dirk, who works in a research group for some company in Germany, and Claudia, who is the group leader and his boss. Both Dirk and Claudia work on a project called Torque.

Organizing Pictures (*Annotation*). Dirk just got back from his holidays in Norway, where he took a lot of pictures. Using his Semantic Desktop, he now wants to organize them, so that he can later find the pictures he wants to easier, generate photo albums for particular topics, etc. A lot of the important concepts probably already exist on his desktop, such as Norway, or the cities he has visited: **Oslo**, **Bergen** and **Trondheim**. Other concepts will be added by Dirk himself, such as **Holidays-in-Norway-2007** and tourist sights like **Preikestolen** or **Holmenkollen**. Since these concepts are more than just tags, Dirk can also say things *about* them, e.g., that **Holidays-in-Norway-2007** was a **Trip** and took place in **2007**, and that **Preikestolen** is a **Location** in **Norway**. Dirk even managed to take a picture of Prince Håkon and Princess Mette-Marit, so he creates two more concepts **Håkon** and

³ Within the NEPOMUK Project, these personas were created by distilling typical users from a series of interviews and evaluations with our use-case partners.

Mette-Marit. There are many ways in which Dirk can link (or tag) his pictures to the relevant concepts – however, part of the Semantic Desktop are intuitive user interfaces, which hide most of the intricacies that go on under the hood from the user. E.g., Dirk might have an application that shows all the concepts that he is interested in the form of a tag cloud. Linking the pictures would then simply require him to drag them onto the desired concept in the cloud.

Planning a Trip (*Context*). Later, Dirk finds out that he has to go on a work trip: a conference in Oslo. The Semantic Desktop assists him in planning and organizing this trip, through the notion of *context*. Dirk can create a new **Trip** object **Trip-to-OOC2007-Oslo** and tell his desktop that he is now in the context of this trip. This means that everything he does from this moment on will be interpreted as happening in that context, until he quits the context again. When he books a flight in his Web browser, the destination field will automatically be filled in with “Oslo”, similarly the departure field. Afterwards, when he books a hotel room, he will be assisted similarly. Dirk will receive a number of email confirmations, such as the flight itinerary and booking confirmation for his hotel. These emails and their attachments will automatically be filed as belonging to the **Trip-to-OOC2007-Oslo** context, so that Dirk can easily find them again later. Once he knows his exact flight dates and where his hotel will be, he enters this information into his calendar, which is also context-aware and will therefore remember that these entries belong to Dirk's trip.

The Social Dimension

Users will have a lot of benefit from just using the Semantic Desktop on their own. However, by connecting to others, a number of additional possibilities arise.

Assigning Tasks in a Group (*Social Interaction*). In the previous scenario, Dirk found out he had to go on a business trip. In fact, he found out about this because he was notified by his boss Claudia, who also uses a Semantic Desktop. Claudia plans to travel to the OOC2007 conference in Oslo to present a research prototype her group has developed as part of the *Torque* project. She does not want to travel alone, so she first needs to find out who of her group members are available while the conference runs. Through the network of Social Semantic Desktops, her calendar application has access to the calendars (or parts of them) of all her contacts. She can ask the calendar to give her a list of all people in her group (**My-Research-Group**) who are working on the *Torque* project (**Torque-Project**) and are free when OOC2007 is on. She finds out that Dirk is free at the desired time. Just like Dirk in the previous scenario, she creates a **Trip-to-OOC2007-Oslo** object and makes it her current context. She also links the trip to the **Torque-Project**. Now, she creates a new **Task** object **Dirk-Prepare-Trip-To-OOC2007**, with a subtask **Dirk-Prepare-Presentation-Slides** and afterwards sends an email to Dirk, asking him to accompany her to the conference, book flights and hotel rooms, and prepare slides for the conference presentation. Her email and the task will of course be

automatically linked to the proper context. Also, in this version of the scenario, Dirk no longer has to create the **Trip-to-OOC2007-Oslo** object himself – instead, it will be added to his Semantic Desktop automatically when he gets Claudia's mail.

FUNCTIONALITIES

In this section we describe a list of functionalities that are needed to support the scenarios mentioned above, as well as other scenarios developed during the NEPOMUK project. The SSD is a platform used to develop different kinds of social and semantic applications. These applications share common functionalities which must be supported by the SSD. We have divided them into five groups, which can be considered different aspects of the SSD. Tab. 1 shows the five different aspects and the individual functionalities within each group. Below we briefly describe the use of each functionality.

Desktop	Annotation, Offline Access, Desktop Sharing, Resource Management, Application Integration, Notification Management
Search	Search, Find Related Items
Social	Social Interaction, Resource Sharing, Access Rights Management, Publish/Subscribe, User Group Management
Profiling	Training, Tailor, Trust, Logging
Data Analysis	Reasoning, Keyword Extraction, Sorting and Grouping

Tab. 1. Functionalities of the Social Semantic Desktop

Desktop. At the *desktop* level, the semantic functionality common to most applications is the ability to add information about any resource. **Annotation** comprises the facilities to store and retrieve semantic relations about anything on the desktop. When Dirk annotates his photos from his trip, he does it from his most favorite photo application (such as Picasa or iPhoto), the annotations are then stored by the SSD. We name this functionality **Application Integration**; applications interact with the SSD by means of different services. When Dirk got notified about the trip to Oslo, this was an example of **Notification Management**. The SSD handles different kinds of mechanisms such as emails, RSS, or text messaging. When Dirk creates a new concept or even a new file on the SSD, the application he uses interacts with the **Resource Management** facilities of the SSD, creating the needed semantics according to the current context and setup. Some of the information Dirk needs when booking his trip are stored on Claudia's desktop. If she is not connected to the network, the **Offline Access** facility exports the relevant

information to another desktop. **Desktop Sharing** is the ability for different users of the SSD to work on the same resources. Claudia might write a report of the trip together with Dirk: the resource management is done on Dirk's desktop, but Claudia can access and edit it remotely.

Search. The semantic network created on the desktop unleashes a whole new way of searching on the SSD. **Search** uses the semantic relations as well as social relations to retrieve relevant items. Once an item is found a user can also **Find Related Items**. For instance, when Dirk searches for a flight to Oslo, he can also search for related items and may find out that another company is actually cheaper, based on the experience of his social contacts.

Social. The SSD provides different means of **Social Interaction**, e.g., the embedding of semantic information in emails or text messaging, or the ability to annotate another user's resources. Some desktop level functionalities such as desktop sharing and offline access require the SSD to enable **Resource Sharing**. When Dirk and Claudia collaborate on the trip's report, Dirk might make it accessible to the whole group by adding it to a shared information space. When sharing resources or information on the network, the **Access Rights Management** of the SSD provides ways to define specific rights relations between users, groups and resources. The SSD's **User Group Management** system makes it easy for the rapid creation of new groups from a list of users. These groups can then be used to modify access rights or for resource sharing in a shared information space. E.g., some of Dirk's friends may have subscribed to get notifications of new pictures that Dirk annotates and makes available. The **Publish/Subscribe** mechanism of the SSD facilitates the creation of feeds of relevant information.

Profiling. If enabled, the **Logging** functionality of SSD logs user activity, which may help to detect the current user's context. The *profiling* of the SSD can be done automatically by **Training**: the SSD learns to predict the user's behavior. The user can still **Tailor** the SSD's intelligent behaviors: some learned contexts can become irrelevant and may need to be re-adapted or removed. The notion of **Trust** on the SSD between people or information sources is also a result of the profiling of the desktop. Dirk might define that he trusts Claudia's information, or Claudia's SSD might learn that Dirk is a trustworthy source of information regarding the *Torque* project.

Data Analysis. To support the training behaviors of the SSD or querying related items, the SSD provides different *data analysis* mechanisms such as **Reasoning**. For instance, when Dirk tags a picture with **Preikestolen** and **Norway**, the SSD may infer that Preikestolen is in Norway. This information can later be reused for search. **Sorting and Grouping** supports applications that perform search. The SSD returns items from many sources and people and sorts and groups these items regarding different criteria, using the semantics defined on these resources. The **Keyword Extraction** from resources such as text resources is useful for automatically tagging or summarizing.

ONTOLOGIES

Ontologies form a central pillar in Semantic Desktop systems, as they are used to model the environment and domain of the applications. The common definition of an ontology is “a formal, explicit specification of a shared conceptualization” (Gruber, 1995)

We distinguish four levels of ontologies for the SSD: *Representational, Upper-Level, Mid-Level and Domain*. The main motivation for having these layers is that ontologies at the foundational levels can be more stable, reducing the maintenance effort for systems committed to using them. A core principle of the Semantic Desktop is that ontologies are used for personal knowledge management. Each user is free to create new concepts or modify existing ones for his *Personal Information Model*. This modeling takes place on the domain-ontology level, but the user is of course free to copy concepts from the other layers and modify them to fit his or hers own needs. In order of decreasing generality and stability the four layers are:

Representation(al) Ontology. Representational ontologies (i.e., ontology definition languages) define the vocabulary with which the other ontologies are represented; examples are RDFS and OWL. The relationship of a representational ontology to the other ontologies is quite special: while upper-level ontologies generalize mid-level ontologies, which in turn generalize domain ontologies, all these ontologies can be understood as instances of the representational ontology. Concepts that might occur in the Representational Ontology level include: classes, properties, constraints, etc.

Upper-Level Ontology. “An upper ontology [...] is a high-level, domain-independent ontology, providing a framework by which disparate systems may utilize a common knowledge base and from which more domain-specific ontologies may be derived. The concepts expressed in such an ontology are intended to be basic and universal concepts to ensure generality and expressivity for a wide area of domains. An upper ontology is often characterized as representing common sense concepts, i.e., those that are basic for human understanding of the world. Thus, an upper ontology is limited to concepts that are meta, generic, abstract and philosophical. Standard upper ontologies are also sometimes referred to as foundational ontologies or universal ontologies.” (Semy et. al, 2004) In the upper-level ontology you will find concepts like: **Person, Organization, Process, Event, Time, Location, Collection**, etc.

Mid-Level Ontology. “A mid-level ontology serves as a bridge between abstract concepts defined in the upper ontology and low-level domain specific concepts specified in a domain ontology. While ontologies may be mapped to one another at any level, the mid-level and upper ontologies are intended to provide a mechanism to make this mapping of concepts across domains easier. Mid-level ontologies may provide more concrete representations of abstract concepts found in the upper ontology. These commonly used ontologies are sometimes referred to as utility ontologies.” (Semy et. al, 2004). The mid-level ontologies may include concepts such as: **Company, Employer, Employee, Meeting**, etc.

Domain Ontology. “A domain ontology specifies concepts particular to a domain of interest and represents those concepts and their relationships from a domain specific perspective. While the same concept may exist in multiple domains, the representations

may widely vary due to the differing domain contexts and assumptions. Domain ontologies may be composed by importing mid-level ontologies. They may also extend concepts defined in mid-level or upper ontologies. Reusing well established ontologies in the development of a domain ontology allows one to take advantage of the semantic richness of the relevant concepts and logic already built into the reused ontology. The intended use of upper ontologies is for key concepts expressed in a domain ontology to be derived from, or mapped to, concepts in an upper-level ontology. Mid-level ontologies may be used in the mapping as well. In this way ontologies may provide a web of meaning with semantic decomposition of concepts. Using common mid-level and upper ontologies is intended to ease the process of integrating or mapping domain ontologies.” (Semy et. al, 2004). Domain ontologies consist of concepts like: **Group Leader, Software Engineer, Executive Committee Meeting, Business Trip, Conference**, etc.

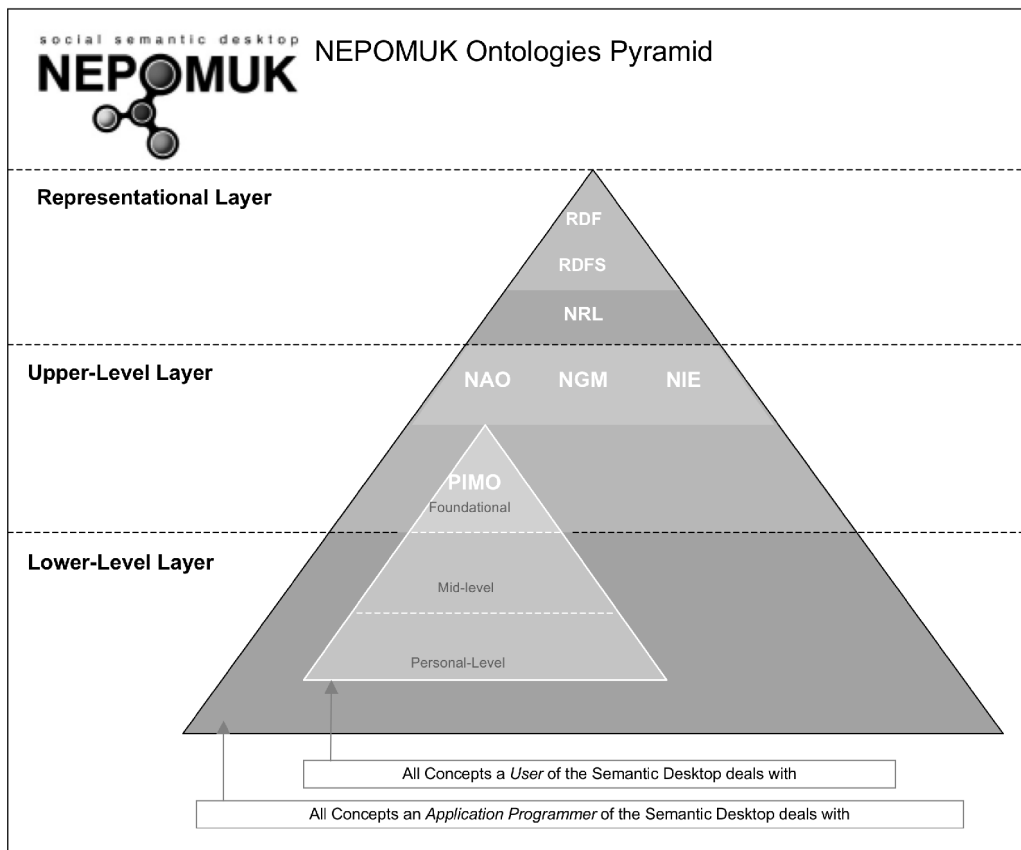


Fig. 3. NEPOMUK Ontology Pyramid

Fig. 3 shows how these four layers relate to the four ontologies created and used in the NEPOMUK Project. As detailed in Sect. 6, we were hesitant to make use of OWL for the representational ontology level in NEPOMUK, and in its place we developed the NEPOMUK Representational Language (Sintek et. al, 2007) (NRL). NRL defines an extension to the semantics offered by RDF and RDFS; the main contribution of NRL is

the formalization of the semantics of named graphs. NRL allows multiple semantics (such as open and closed world) to coexist in the same application, by allowing each named graph to have separate semantics. The NEPOMUK Annotation Ontology (NAO) is a basic schema for describing annotations of resources, this is essentially a formalization of the tagging paradigm of Web2.0 applications. A specialized part of NAO is the NEPOMUK Graph Metadata schema (NGM) which allows the description of named graphs, defining meta-data properties such as the author, modification dates and version data.

Finally, the NEPOMUK Information Elements ontology (NIE) contains classes and properties for describing objects found on the traditional desktop, such as files (Word documents, images, PDFs), address book entries, emails, etc. NIE is based on existing formats for file meta-data such as EXIF for image meta-data, MPEG7 for multimedia annotations, ID3 for music files, iCal, and others.

TECHNOLOGY

The Social Semantic Desktop deploys the Semantic Web on the desktop computer. Therefore, the technology stack proposed for the Semantic Web (the famous “Layercake”⁴ shown in Fig. 4) is adopted for the SSD as well.

However, there are some specific considerations for the desktop scenario: everything on the desktop should be identifiable by URIs. This is partially solved for files, where RFC1738⁵ specifies the form of file:// URIs, but requires considerable care for other applications which may not represent their data entities as individual files, such as address books or email clients.

Secondly, one can note that for the single desktop scenario there are fewer requirements on aspects such as trust, proof, and signatures. When one progresses to the Social Semantic Desktop, which involves interactions between many users, these aspects must be considered again.

⁴ Tim Berners-Lee talk, XML and the Web: <http://www.w3.org/2000/Talks/0906-xmlweb-tbl/>

⁵ RFC1738: <http://tools.ietf.org/html/rfc1738>

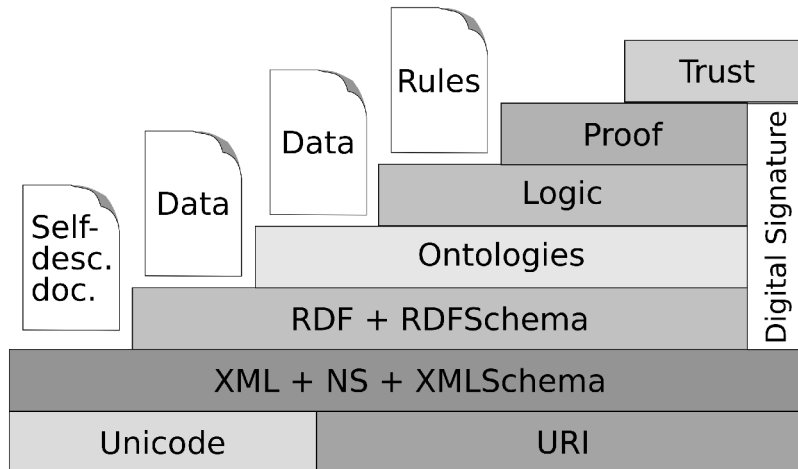


Fig. 4. The Semantic Web Technology Stack

In NEPOMUK we chose not to use the Web Ontology Language (OWL) (McGuinness et al, 2004) as an ontology language, because of the challenge of dealing with (and implementing) OWL correctly; because our ontological modeling requirements were modest, and, most importantly, because OWL enforces an open-world view of the world, which did not seem to be appropriate for the (local) Semantic Desktop. In a World Wide Web context it is impossible for an application to read all available data, and an open-world assumption is natural, since additional data can be discovered at any moment. However, the open-world assumption makes it impossible to adopt negation as failure (Clark, 1978) which makes practical application development difficult and is also difficult to explain to the end-user. In the context of a local desktop application the situation is quite different, here it is perfectly possible to read all data available, and the closed world assumption makes much more sense. In place of OWL we developed our own ontology specification language called NRL, which uses the closed-world assumption. An additional RDF-based technology that we use widely, but which does not feature in the Semantic Web stack is the concept of named graphs (Carroll et. al, 2005). This allows one to divide a larger RDF store into sets of RDF statements (graphs), where each is identified with a URI (the name). In this way it is possible to make meta-statements about each graph, such as provenance information. Named graphs thus become an alternative to RDF reification, which also allows making statements about other statements, but is harder to implement and creates a significant overhead. NRL does also allow applying different semantics for different named graphs, thus allowing us to integrate the local closed-world with the open-world of the extended Semantic Web.

As noted previously, applications on the Semantic Desktop are analogous to services available on the Web. Each application will offer an interface for exposing the functionality it offers. Although a single desktop is not distributed, a network of SSDs is. It therefore suggests itself to adopt the Web Service stack of tools for inter-service communication for the Semantic Desktop: the Web Service Description Language (WSDL)⁶ which is used for describing the interfaces of services offered, XML Schema

⁶ WSDL: <http://www.w3.org/TR/wsdl>

(XSD)⁷ which is used for primitive type definitions, and finally the Simple Object Access Protocol (SOAP)⁸ which is used for the invocation of services. In Sect. 8 we give further details on how these technologies work in relation to the Semantic Web technologies presented above.

ARCHITECTURE

In our vision, the standard architecture comprises a small set of standard interfaces which allow application developers to easily extend it and ultimately lead to an evolving ecosystem. Fig. 5 depicts this set of interfaces transposed into services, together with their placement in the NEPOMUK architecture. The architecture has to reflect the two aspects of the scenarios introduced in Sect. 3, i.e., the semantic (which can operate on a single desktop) and the social aspect (which is relevant in a network of desktops). To cover these requirements and the functionalities discussed in Sect. 4, the SSD is organized as a Service Oriented Architecture (SOA). Each service has a well defined WSDL interface and is registered at the *Service Registry*. The social aspect of sharing resources over the network is enabled by the peer-to-peer (P2P) infrastructure of the architecture. In the following we present the services of the SSD.

The architecture, as show in Fig. 5, is organized in three layers. Like current desktop systems, the desktop environment builds on top of the *Operating System* core, such as the file system, kernel, and network environment. On the SSD the desktop environment is pooled in the *Social Semantic Desktop Middleware Layer* (SSD Middleware). The SSD Middleware groups the services of the SSD to be used in the Presentation Layer, which provides the user with SSD enabled applications that take advantages of the functionalities of the SSD.

The SSD is made up by individual desktops, which are organized in a P2P fashion. To support the communication between the peers, the SSD Middleware provides *P2P Network communication Services*. To enable information sharing between individual desktops, the RDF metadata of shared resources is stored in the *distributed index* of the P2P system. In NEPOMUK, the P2P system is based on GridVine (Aberer et. al, 2004), which in turn is built on top of P-Grid (Aberer et. al, 2003) and provides a distributed index with RDQL query supports.

⁷ XML Schema: <http://www.w3.org/XML/Schema>

⁸ SOAP: <http://www.w3.org/TR/soap>

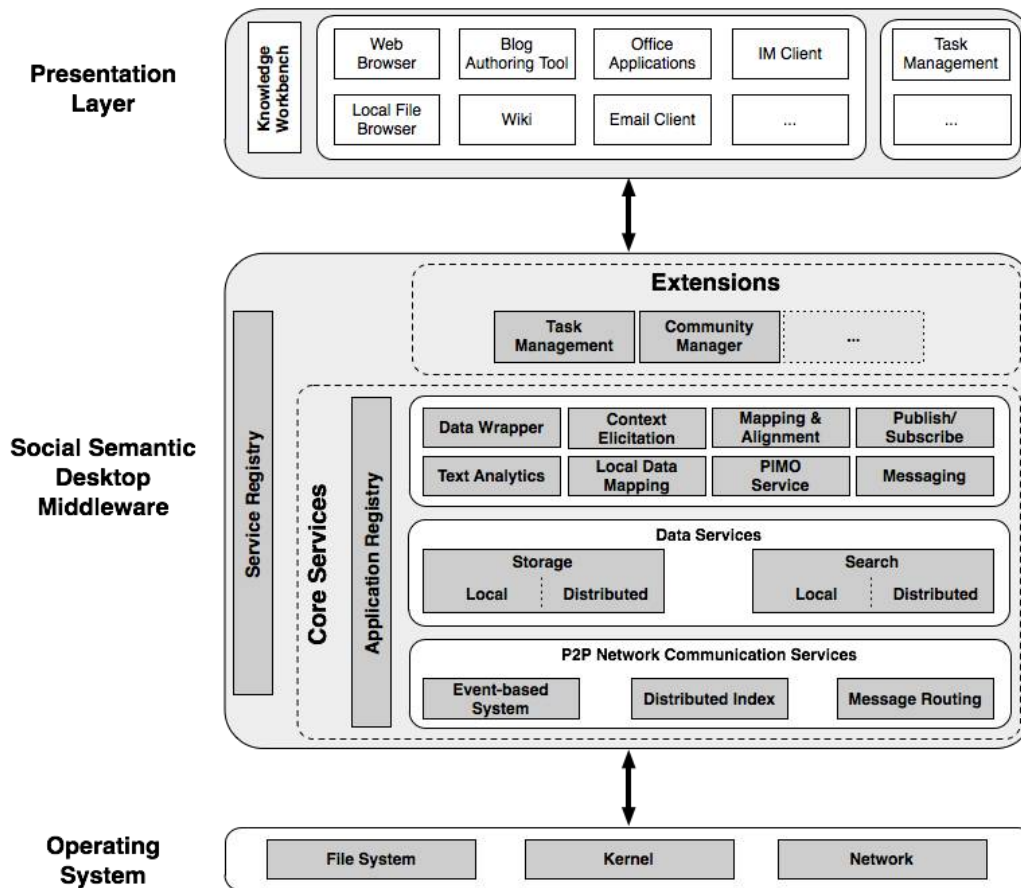


Fig. 5. Layered Architecture of the Social Semantic Desktop

Network Communication Services provide an *Event-based System*, which is responsible for the distribution of the events between the SSD peers. On the SSD, the event-based system is used to support the publish/subscribe system. Users as well as other services can use RDF to describe the kind of information they are interested in (e.g., new pictures of Norway become available, the status of a document changes to final, etc.). These subscriptions are stored in the distributed index of the P2P system. An event that was fired carries an RDF query as payload, which is matched against all subscriptions and triggers the notification of the subscriber. In addition, the *Messaging Routing* system uses RDF information to route messages to receiver.

The *Data Services* are responsible to control the insertion, modification, deletion, and retrieval of resources on the SSD. 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*. Resources and their RDF descriptions can either be added to the SSD manually, or the *Data Wrapper* or *Text Analysis* service extracts the information from desktop applications such as email clients or calendars. Data Wrappers are used to extract

metadata form structured data sources (e.g., email headers, calendar entries, etc.). In NEPOMUK, data wrappers are implemented based on Aperture (Aperture, 2007). The Text Analysis service is used to extract metadata from unformatted text (e.g., email bodies, word processing documents, etc.). For local queries and for offline working the RDF metadata 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 peer-to-peer 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.

Before new metadata can be added to the repository, we have to check whether this metadata describes resources that are already instantiated (i.e., an URI has been assigned) in the RDF repository. In this case, the URI of the resource should be reused, rather than creating a new one. This process is known as information integration (Bergamaschi et. al, 2001). The *Local Data Mapper* service takes over this responsibility in the SSD Middleware. E.g., the Data Wrapping service extracts contact information from the address book and stores the metadata in the repository. Since this is the first time information about the contacts is added to the RDF repository, a new URI is generated for each person. If later the Data Wrapping service extracts information from an email header, the Local Data Mapping service is responsible to lookup whether information about the sender of the email is already in the repository and reuse the corresponding URI instead of creating a new one (Sauermann et. al, 2006).

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, or different personal information models). Since individual users share information over the SSD, it is likely to happen that they use different ontologies for their annotations even when talking about similar domains. Therefore, the SSD Middleware provides a *Mapping & Alignment* 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.

The SSD Middleware logs the actions a user performs on the resources on his desktop. The logged data is stored in the Local Storage and analyzed by the *Context Elicitation* service to capture the current working context of the user. The context can for example be used to adapt the user interface or to suggest meaningful annotations to the users, depending on the task they are currently working on.

As discussed in Sect. 6, the services on the SSD use RDF to exchange data. Therefore, services need the capability to generate and process RDF graphs. To simplify the handling of the RDF graphs, the *Ontology Service* provides an easy way to create and manipulate concepts in RDF graphs.

The *Publish/Subscribe System* allows users or other SSD services to subscribe to events on the SSD. The subscriptions are stored as RDF graphs in the distributed index. If an event occurs, the RDF query of the event is matched against the subscriptions. When the subscription, i.e., the RDF query, matches the event, the *Messaging* service 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 SSD users.

The *Core Services* of the SSD Middleware comprise the services which provide the basic functionalities of the SSD. These services can be accessed via the SSD Application Programming Interface (API). If a developer wants to exploit the SSD Core Services to build his domain-specific application, he can do this as an *extension* of the SSD Middleware. An example for such an extension is the *Task Management* which provides functionalities such as creating, delegating, and manipulating of tasks. Finally, the *Application registry* allows applications from the Presentation Layer to register call back methods at the SSD Middleware if they need to be notified by SSD services, e.g., when a message arrives and has to be displayed to the user in an Instant Messaging Client.

The top layer of the architecture is the presentation layer. It provides a user interface to the services provided by the SSD, and is built using the SSD API. Many desktop applications are possible sources for resources that should be managed by the SSD. Therefore, each desktop application should integrate support for the SSD Middleware. Since this assumption does not hold for most of the current off-the-shelf applications, we developed plug-ins and add-ons to enable a seamless integration with existing applications. These plugins for example extract email or calendar data and add them as resources to the SSD. However, within NEPOMUK we also develop dedicated applications that make use of the SSD API directly, such as a semantic *Wiki* or *Blogging Tools*. (Möller et. al, 2006)

In addition, the *Knowledge Workbench* is the central place to browse, query, view, and edit resources and their metadata. In this way the Knowledge Workbench aims to replace current file management tools such as the MS File Explorer. If the SSD is extended by usage extensions, the application programmer also has to provide the corresponding user interface in the Presentation Layer (e.g., for Task Management, Community Management, etc.).

IMPLEMENTATION AND ENGINEERING PRINCIPLES

As detailed in Sect. 7. we deem a Service Oriented Architecture (SOA) to be most suitable for the SSD framework. Furthermore, we decided to use the industry standard SOAP (Simple Object Access Protocol) for exchanging messages between our components. For traditional applications the names and structure of SOAP messages is specified using the Web Service Description Language (WSDL), which in turn uses XML schema data-types to specify the form of the objects being exchanged. However, since the formal modeling of the target domain using ontologies is the core idea of a Semantic Desktop application, the best-practices for SOAs are slightly different. In this section we will discuss some important differences from a traditional SOA system.⁹ Basing a system architecture on underlying domain ontologies is similar in nature to Model Driven Architectures (MDA)¹⁰. However, on the SSD, ontologies take the place of UML models.

⁹ In this chapter we make the assumption that a modern object-oriented programming language like Java will be used for implementation, but observations and solutions are equally valid for most other languages.

¹⁰ MDA: <http://www.omg.org/mda/>

Working with RDF

Sect. 5 described the substantial effort that went into the modeling of our domains as ontologies in a formal language. These ontologies give us a very powerful and flexible modeling language, although the structure of instances of such ontologies at first sight seem much more constrained than complex XML schema data-types, the flexibility of RDF introduces some additional requirements for developers of components that should handle RDF instances:

- The structure of the RDF instances received may not be fully known at design time. This means one must take great care that the code does not break when encountering unknown properties in the data, and these unknown properties must also be preserved. In general, programming services for the Semantic Desktop is more like programming services for the web, rather than for traditional desktop applications, and one should follow the general rule of web-programming: “Be strict in what you send and tolerant in what you receive.”
- Conversely, other services might not be aware of all the properties the local service uses. Therefore each service must be programmed to be tolerant of missing data and do their best with the data that was provided.

Passing Instances in Messages

Normally, when using SOAP in connection with WSDL and XML schema for data modeling, some mapping is used that will convert the XML schema definition to class definitions in the programming language of choice. Furthermore, stubs and skeletons will be generated for the service themselves, so that the details of communication are hidden. Programming against remote services is then indistinguishable from programming against a local object. However, when using services that pass instances for which the structure is defined by ontologies, the mapping is not so straight forward. Although interaction with RDF data can always be done on a completely general level using basic RDF APIs we are interested in facilitating the job of programmers consuming our services, and allowing them to work on a higher level than RDF triples. We identify three alternatives for programming web services where parameters are instances from an ontology:

- Starting with the ontologies, a number of tools¹¹ can be used to create a set of Java classes from the ontologies. The service interface is written using parameters of these types, and another tool is used to generate the WSDL and associated XML schema types from these. By sharing the URIs of the concepts in the ontologies with the URIs of the XML schema types, the semantics of messages is retained. The benefit of this approach is that much of the SOAP technology is retained, existing tools may be reused. Also, developers who are not familiar with Semantic Web technology will find that developing and using these services is unchanged from a normal Java environment. The main problem with this approach comes

¹¹ RDFReactor: <http://wiki.ontoworld.org/wiki/RDFReactor>; RDF2Java: <http://rdf2java.opendfki.de>; Elmo: <http://openrdf.org>, etc.

from the fact that ontologies are in general more dynamic than Java class definitions. In particular, as noted in Sect. 5, we expect the personal information models to change frequently. This approach requires a complete re-run of the whole tool chain and a recompile of the system when an ontology changes, as well as introducing some constraints on the ontologies.

- On the other end of the spectrum it is possible to bypass the parameter passing of SOAP all together, and rely more on the Semantic Web technology. Each method offered by a service will take a single RDF document (possibly including several named-graphs), and all the details about the message are given in these RDF graphs. An additional ontology for messages and parameters must be constructed, and some named-graph aware serialization (e.g., TriG or TriX¹²) of RDF is used to construct the XML SOAP messages. This approach was, for instance, used in the SmartWeb project¹³. The benefit of this approach is that the effort that has gone into modeling the ontologies is not duplicated for modeling objects. Also, the full expressivity of RDF may be used when modeling, as it not required that the instances fit into another representation. The backside to this flexibility is that it is significantly harder to program with RDF graphs than with simple Java objects, and both service developers and consumers need good knowledge about RDF. One can of course envisage new tools that facilitate programming with such RDF messages, but since all the interesting details are hidden inside the RDF parameter, existing SOAP tools for development or debugging are no longer very useful.
- Finally, a hybrid approach of the two methods is possible. Here each method will retain multiple arguments, but each argument is represented by an RDF resource. We envisage two possibilities for doing this: either each parameter is given as a (*named-graph-uri, uri*) tuple pointing into an RDF document given as a special parameter; or, alternatively, each parameter is in itself an RDF graph plus the URI of the actual parameter (each RDF graph may contain several resources). The benefit of this method is that the changes in the ontology do no longer require a recompile of the system, while at the same time allowing slightly more compatibility with existing SOAP tools. The problem with this method remains that both client and server programmers need in-depth knowledge of RDF and the ontologies used.

Regardless of which of the three alternatives one chooses, it remains an important issue to make sure that the formal description of the services (i.e., the WSDL+XML Schema definitions) remain semantically correct and retain the pointers to the ontology concepts which the parameters represent. As mentioned, for the first approach this can be handled by well chosen URIs for the XMLSchema types. For the second and third approach the parameters have the form of simple string objects in both the WSDL definition and the SOAP messages, since the RDF serialization is represented as a string. However, both versions of WSDL available at the time of writing allow extensions to the WSDL format itself¹⁴, and additional constraints on the type or form of the RDF instances contained inside the string parameters may be specified here. This is the approach taken by the

¹² TriG/TriX: <http://www.w3.org/2004/03/trix/>

¹³ SmartWeb: <http://www.smartweb-project.de/>

Semantic Annotation for WSDL and XML Schema (SAWSDL) working group¹⁵ and the NEPOMUK project makes use of their standard.

In this section we have considered a very lightweight approach to semantically enriching SOAP Web Services by passing RDF-based parameters. If a more powerful approach is required, the reader is advised to look into OWL-S¹⁶ and the Web Service Modeling Language (WSML)¹⁷, both defining much more sophisticated frameworks for Semantic Web Services.

RELATED WORK

In the following we review relevant research and development approaches for the Social Semantic Desktop. After providing a brief description, we discuss the lessons learned and state our conclusions.

Gnowsis (Sauer mann, 2003) was among the first research projects targeting a Semantic Desktop system. Its goal is to complement established desktop applications and the desktop operating system with Semantic Web features, rather than replacing them. The primary focus of Gnowsis is on *Personal Information Management* (PIM). It also addresses the issues of identification and representation of desktop resources in a unified RDF graph. Gnowsis uses a Service Oriented Architecture (SOA), where each component defines a certain interface and it is available as an XML/RPC service.

The **Haystack** (Quan et. al, 2003) project presents a good example for an integrated approach to the SSD field. Inter-application barriers are avoided by simply replacing these applications with Haystack's own word processor, email client, image manipulation, instant messaging, etc. Haystack allows users to define their own arrangements and connections between views of information, thus making it easier to find information located in the personal space. The Haystack architecture can be split into two distinct parts: the Haystack Data Model (HDM) and the Haystack Service Model (HSM). The Data Model is the means by which the user's information space is represented, similar to what has been discussed in Sect. 5. The set of functionalities within Haystack is implemented by objects in the Haystack Service Model (HSM). Haystack has a standard three-tiered architecture, consisting of a user interface layer (the client), a server/service layer, and a database. Haystack was ground-breaking in terms of the dynamic creation of user interfaces, but the project ended before establishing any standards.

Another relevant personal information management tool is the **Semex System** (SEMantic EXplorer) (Dong et. al, 2005). Like other tools, it organizes data according to a domain ontology that offers a set of classes, objects and relationships. Semex leverages the Personal Information Management (PIM) environment to support on-the-fly integration of personal and public data. Information sources are related to the ontology through a set

¹⁴ Language Extensibility in WSDL1: http://www.w3.org/TR/wsd1#_language and in WSDL2: <http://www.w3.org/TR/wsd20#language-extensibility>

¹⁵ SAWSDL: <http://www.w3.org/TR/sawSDL/>

¹⁶ OWL-S: <http://www.daml.org/services/owl-s/>

¹⁷ WSML: <http://www.wsmo.org/wsml/>

of mappings. Domain models can be shared with other users in order to increase the coverage of their information space. When users are faced with an information integration task, Semex aids them by trying to leverage data collected from previous tasks performed by the user or by others. Hence, the effort expended by one user later benefits others. Semex begins by extracting data from multiple sources and for these extractions it creates instances of classes in the domain model. It employs multiple modules for extracting associations, as well as allowing associations to be given by external sources or to be defined as views over other sets of associations. To combine all these associations seamlessly, Semex automatically reconciles multiple references to the same real-world object. The user browses and queries all this information through the domain model.

A similar idea is exploited by the **IRIS** Semantic Desktop (Cheyer et. al, 2005) (“Integrate. Relate. Infer. Share”), an application framework that enables users to create a “personal map” across their office-related information objects. IRIS offers integration services at three levels:

- Information resources (e.g., email messages, calendar appointments) and applications that create and manipulate them must be accessible to IRIS for instrumentation, automation, and query. IRIS offers a plug-in framework, in the style of the Eclipse architecture, where “applications” and “services” can be defined and integrated within IRIS. Apart from a very small, lightweight kernel, all functionality within IRIS is defined using a plug-in framework, including user interface, applications, back end persistence store, learning modules, harvesters, etc. Like Haystack, inter-application barriers do not exist, because all applications are made from scratch for IRIS.
- A Knowledge Base provides the unified data model, persistence store, and query mechanisms across the information resources and semantic relations among them. 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 notification pane, menu toolbar management, query interfaces, the link manager, and suggestion pane.

DeepaMehta (Richter et. al, 2005) is an open source Semantic Desktop application based on the Topic Maps standard¹⁸. The DeepaMehta UI, which runs through a Web browser, renders Topic Maps as a graph, similar to concept maps. Information of any kind as well as relations between information items can be displayed and edited in the same space. The user is no longer confronted with files and programs. DeepaMehta has a layered, service oriented architecture. The main layer is the application layer, which offers various ways for the presentation layer to communicate with it via the communication layer (API, XML Topic Maps (XTM) export, EJB, SOAP). Finally, the storage layer holds all topics and their data either in a relational database or simply in the file system.

Other relevant projects include **Beagle++** (Brunkhorst et. al, 2005), a semantic search engine which provides the means for creating and retrieving relational metadata between information elements present on the desktop, **DBIN** (Tummarello et. al, 2006), which is similar to a file sharing client and connects directly to other peers, **PHLAT** (Cutrell et. al,

¹⁸ ISO/EIC 13250:2003: <http://www.y12.doe.gov/sgml/sc34/document/0129.pdf>

2006), a new interface for Windows, enabling users to easily specify queries and filters, attempting to integrate search and browse in one intuitive interface, or **MindRaider**¹⁹, a Semantic Web outliner, trying to connect the tradition of outline editors with emerging SW technologies. The **MyLifeBits** project by Microsoft Research is a lifetime store of multimedia data. Though the system does not intent to be a SSD, one can learn from it how to integrate data, i.e., how to manage the huge amount of media and how to classify/retrieve the data (Gemmell et. al, 2002). It combines different approaches from HCI (Computer-Human Interaction) and information integration, while it lacks a conceptual layer beyond files. The **Apogée**²⁰ project deals with data integration in applications related to Enterprise Development Process (ECM). It aims at building a framework to create Enterprise Development Process-oriented desktop applications, independent from vendor or technologies. Finally, starting from the idea that everything has to do with everything, has a relationship with everything, **Fenfire**²¹ is a Free Software project developing a computing environment in which you can express such relationships and benefit from them.

Although the systems we have looked at focus on isolated and complementary aspects, they clearly influenced the vision of the SSD presented here. Some of the architectural decisions made in the NEPOMUK project and presented in this chapter are similar to those of platforms like Haystack, IRIS, and DeepaMetha, e.g., in that we present a User Interface Layer, a Service and a Data Storage Layer. The modular architecture, also identified within the Haystack, SEMEX, and DeepaMetha systems, as well as the standardized APIs offer an easy way of introducing new components. Our approach guarantees that each component may be changed without affecting other components it interacts with. The interaction has to suffer only in the case in which the API of the component is modified. The NEPOMUK Architecture also provides service discovery functionalities: the NEPOMUK Registry providing a proper support for publishing and discovering the existing NEPOMUK Services by using a standard interface.

CONCLUSION

We presented the Social Semantic Desktop as a comprehensive approach to information handling. Oriented at the needs of knowledge workers, this approach centers around supporting the main information-oriented activities: The articulation of knowledge and the generation of new information items; the structuring, relating, and organization of information, and the sharing of formal and informal information within networks of cooperating people. From this, we derived key functionalities of the desktop, but also for search, social interaction, profile building, and data analysis.

Building the SSD relies on basic principles: Whatever appears within the personal workspace is treated as an information item. Content, relations, special services all refer to formal annotations of such information items, which in turn link between information items and personal information models. Unifying the flexibility and personal liberty of expressing whatever concepts seem relevant with the commitment to socially shared

¹⁹ MindRaider: <http://mindraider.sourceforge.org/>

²⁰ Apogée: <http://apogee.nuxeo.org/>

²¹ Fenfire: <http://www.fenfire.org/>

conceptualizations results in a layered hierarchy of ontologies which allow the necessary differences in stability, sharing scope, and formality. Integrating the tools of everyday information processing asks for an easy and flexible integration of existing desktop applications. Finally, the adoption of Semantic Web standard technology for representation and communication enables the easy transgression from personal annotated information to shared Semantic Web content.

Consequently, the architecture of the SSD combines standards-based data repositories with a rich middleware, which in particular allows for manifold service integrations and communications. On top of that, various presentation clients and specific applications support whatever activities are performed on the desktop. Such applications may be highly domain-specific, although core functionalities of knowledge work trigger standard applications, e.g., for document processing, task management, communication, etc.

The design decisions presented result in particular implementation and engineering principles; we outlined the adaptation to RDF, the service integration, and the message passing mechanisms in particular.

In summary, the SSD offers the basic technology and tools for everyday information processing by knowledge workers. In order to reach the intended wide acceptance and broad uptake, care was taken to make the central software components available under open source licenses, and to encourage the development and contribution of application-specific enhancements and adaptations. The concept of the SSD is promising and relies on a number of techniques which reach their maturity right now – consequently, a number of research and development projects are under way and contribute to the overall evolution of the concept.

Following the realizations described in this chapter, we see the SSD as a basis for the self-motivated generation of semantically annotated information, which will not only help the individual by allowing multitudes of specific services and support, but will also initiate a wide movement to populate the Semantic Web.

FUTURE RESEARCH DIRECTIONS

Although the ideas of the Social Semantic Desktop are based on solid foundations as presented here, the research areas surrounding this topic are still in their infancies. We will briefly discuss some of the pre-dominant challenges in the coming years:

Trust and Privacy. As pointed out in the Semantic Web technology stack presented earlier, a crucial component for any high-level Semantic Web service is the issue of trust and privacy. Trust touches on a wide range of issues, from the technical issues of cryptographic signatures and encryption, to the social issues of trust in groups and among individuals. These issues are all as valid for the Social Semantic Desktop as for the Semantic Web in general, or perhaps even more so, as people are less critical of putting personal data on their personal desktop.

User, group, and rights management. When a single personal Semantic Desktop allows easy sharing of information with the network of Social Semantic Desktops, determining access rights for this information becomes very important. The Social Semantic Desktop

sets new requirements for distributed authentication, flexible group management, and fine-grained access rights, all the while remaining intuitive and unobtrusive for the end user.

Integration with the wider Semantic Web and Web 2.0. Currently we are talking about the Social Semantic Desktop as a network of Semantic Desktops built on the same standards. It is important to remember that the key benefit of Semantic technology is the easy access to integration with anyone using the same representational languages and ontologies. The growth of feature-rich Web applications is growing rapidly, and ensuring a strong bond between the Semantic Desktop and these services is a continuous challenge.

Ontologies and Intelligent Services. To date ontologies have been used to introduce a common vocabulary for knowledge exchange. On the Social Semantic Desktop ontologies are used to formalize and categorize personal information. This introduces many interesting issues around ontology versioning, ontology mapping, and ontology evolution. Furthermore, using ontologies with well-defined semantics will allow intelligent services to be built (e.g., using reasoning) that allow for much more than just browsing and (simple) searching.

User Evaluation. The underlying thesis of the whole (Social) Semantic Desktop effort is that the added semantics will improve productivity and enable new forms of cooperation and interaction which were not previously possible. In-depth empirical evaluation with real users of a Social Semantic Desktop systems are required to determine if this thesis really holds.

ADDITIONAL READINGS

From a historical perspective, the most important references in the Social Semantic Desktop domain are those by Vannevar Bush (1945) and Doug Engelbart (1962) which we mentioned in Sect. 2. Another important early influence is certainly Ted Nelson's work on hypertext (Nelson, 1965). A modern vision of those ideas is a paper by Decker and Frank (2004), which also coined the term "Semantic Desktop". Of course, any work that is based on the ideas of the Semantic Web is not complete without references to seminal papers such as (Berners-Lee et. al, 2001) or (Hendler, 2001). In fact, the original vision of the World Wide Web itself already contained the idea of an information space that would reach from "mind to mind" (Berners-Lee, 1999); a thought that is central to the SSD.

Current and recent research and development in the SSD domain has already been presented in Sect. 9. However, one influence that has not been covered in this chapter so far, but is closely related to the idea of a Semantic Desktop is the concept of *Semantic File Systems* – file systems in which files are not organized hierarchically, but rather according to their metadata. The concept and an early implementation are described in detail in (Gifford et. al, 2001).

Most of the references given in this chapter are of a technical nature. However, one has to keep in mind that the SSD is a tool for *information management* and *knowledge work*, and thus psychological and sociological research into the nature of knowledge work in any form are relevant as well. Oren (2006) provides a detailed overview of literature in this field, with the intention of applying the lessons learned to the development of the Semantic Desktop.

Finally, as another entry point for additional reading, we would like to point the reader to the series of *Semantic Desktop Workshops* which were co-located with the International Semantic Web Conferences in 2005²² and 2006²³.

QUESTIONS FOR DISCUSSION

Q: I prefer to handle my photo collection in a web 2.0 photo sharing environment. Is this compatible with the Social Semantic Desktop? May I keep the work I have invested here?

A: Yes. Every photo in your collection can be reached via a specific URI, thus it can be handled as a particular information item in the SSD. You might implement a suitable wrapper to transfer local annotations from your SSD onto the photo sharing platform, if you intend to disclose this information.

Q: The Social Semantic Desktop presupposes that everything is an information item. What about entities which are not information but real-world objects? Can I manage them in the SSD and add comments about them, e.g., about my friend's cat?

A: The solution is easy: Just create an appropriate description of the real world object within your SSD, thus creating an URI for the object in question. Let's say you create an instance of the class *pet* in your SSD (assuming you have this category within your SSD) and describe it as 'well-known house cat'. Then you can link this instance to, e.g., a photo of the animal, or you add an 'owns' link which connects it to the URI of your friend, and so on. Making an arbitrary object re-appear as a formal instance within the SSD models is often called 're-birthing', btw.

Q: Think about scenarios you encounter every day, and where the SSD can make your work easier.

A: The answer is of course a personal one, but for a typical knowledge worker (researchers, students, journalists, etc.) here are some example ideas:

- Show me related appointments when composing emails to a person, i.e., You also have lunch with Claudia next week.
- Show me previously viewed PDF documents on the same topic when researching on Wikipedia.

²² SemDesk2005: <http://tinyurl.com/yuxpld>

²³ SemDesk2006: <http://tinyurl.com/2hqfak>

- Remember my meal and window preferences when booking flights.
- Remind me of my previous idea of combining topic A with topic B when reviewing my topic A notes.
- Let me connect an incoming email from a student to the colleague who introduced me to that student.

Q: What are the benefits of the Social Semantic Desktop compared to solution such as Microsoft Exchange server or the tight integration of applications on MacOSX? They also fulfil many of the functionalities required by the scenarios outline in this chapter.

A: The Social Semantic Desktop is different because of the standards used to build it. Firstly, by basing the representational layers of the Semantic Desktop on the existing (Semantic) Web standards we enable interoperability by a wide range of existing projects, and secondly, by creating new standards for desktop integration and data-formats we encourage future software developers to build on top of the Semantic Desktop. On the Semantic Desktop both the applications and the data encourages open access, and this exactly the opposite of the vendor lock-in that for instance Exchange Server aims for.

Q: Inspect the current state of the Semantic Web and the data available. What data-sources and/or ontologies do you think could be useful for integration with the Semantic Desktop?

A: The answer will of course change as the Semantic Web evolves, but at the time of writing relevant ontologies include:

- The Friend-of-a-Friend project – <http://xmlns.com/foaf/spec>
- The Description-of-a-Project schema – <http://usefulinc.com/doap/>
- The Semantically Interlinked Online Communities project – <http://siocproject.org/>
- Dublin Core for basic meta-data – <http://dublincore.org/>

Useful data-sources and/or web-services include:

- GeoNames for (reverse) geocoding – <http://www.geonames.org/>
- DBpedia for a Semantic Web view of Wikipedia – <http://DBpedia.org/>

REFERENCES

Aberer, K., & Cudré-Mauroux, P., & Datta, A., & Despotovic, Z., & Hauswirth, M., & Puceva, M., & Schmidt, R. (2003). P-Grid: A Self-organizing Structured P2P System. *SIGMOD Record*, 32(3):29–33.

Aberer, K., & Cudré-Mauroux, P., & Hauswirth, M., & Pelt, T.V. (2004). Gridvine: Building Internet-Scale Semantic Overlay Networks. In S. A. McIlraith, D. Plexousakis, F. van Harmelen (Eds.), *The Semantic Web – ISWC 2004: Third International Semantic Web Conference*, pp. 107–121. Springer Verlag.

Aperture: A Java Framework for Getting Data and Metadata, Last visited March 2007. <http://aperture.sourceforge.net/>.

Bergamaschi, S., & Castano, S., & Vincini, M., & Beneventano, D. (2001). Semantic Integration and Query of Heterogeneous Information Sources. *Data & Knowledge Engineering*, 36(3):215–249.

Berners-Lee, T., & Fischetti, M. (1999). *Weaving the Web – The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco.

Berners-Lee, T., & Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*.

Brunkhorst, I., & Chirita, P. A., & Costache, S., & Gaugaz, J., & Ioannou, E., & Iofciu, T., & Minack, E., & Nejd, W., & Paiu, R. (2006). *The Beagle++ Toolbox: Towards an Extendable Desktop Search Architecture (Technical report)*, L3S Research Centre, Hannover, Germany.

Bush, V. (1945). *As We May Think*. The Atlantic Monthly, July 1945.

Carroll, J. J., & Bizer, C., & Hayes, P., & Sticker, P. (2005). Named Graphs, Provenance and Trust, In A. Ellis, T. Hagino (Eds.), *WWW 2005: The World Wide Web Conference*, pp. 613-622.

Cheyner, A., & Park, J., & Giuli, R. (2005). IRIS: Integrate. Relate. Infer. Share. In S. Decker, J. Park, D. Quan, L. Sauermaun (Eds.), *Semantic Desktop Workshop at the International Semantic Web Conference*, Galway, Ireland, November 6, volume 175.

Clark, K. L. (1978). Negation as failure. In J. Minker (Ed.), *Logic and Data Bases*, Plenum Press, New York, pages 293–322, 1978.

Cutrell, E., & Robbins, D.C., & Dumais, S.T., & Sarin, R. (2006). Fast, Flexible Filtering with PHLAT – Personal Search and Organization Made Easy. R. E. Grinter, T. Rodden, P. M. Aoki, E. Cutrell, R. Jeffries, G. M. Olson (Eds.), *Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006*, Montréal, Québec, Canada, April 22-27, 2006. ACM 2006, ISBN 1-59593-372-7

Decker, S., & Frank, M. (2004). The Networked Semantic Desktop. In C. Bussler, S. Decker, D. Schwabe, O. Pastor (Eds.), *Proceedings of the WWW2004 Workshop on*

Application Design, Development and Implementation Issues in the Semantic Web, New York, NY, USA, May 18, 2004

Dong, X., & Halevy, A.Y. (2005). A Platform for Personal Information Management and Integration. In M. Stonebraker, G. Weikum, D. DeWitt (Eds.), *Proceedings of 2005 Conference on Innovative Data Systems Research Conference*, pages 119–130

Engelbart, D.C. (1962). *Augmenting Human Intellect: A Conceptual Framework (Summary report)*, Stanford Research Institute (SRI).

Gemmell, J., & Bell, G., & Lueder, R., & Drucker, S., & Wong, C. (2002). MyLifeBits: Fulfilling the Memex vision. In *ACM Multimedia* December 1-6, Juan-les-Pins, France, pages pp. 235–238, 2002.

Gifford, D.K., & Jouvelot, P., & Sheldon, M.A., & O'Toole, J.W. Jr. (1991). Semantic File Systems. In *13th ACM Symposium on Operating Systems Principles*, October 1991.

Gruber, T.R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *International Journal of Human-Computer Studies*, volume 43, pages 907–928, 1995.

Hendler, J. (2001). Agents and the SemanticWeb. *IEEE Intelligent Systems*, 16(2):30–37, March/April 2001.

McGuinness, D.L., & van Harmelen, F. (2004). *OWL Web Ontology Language Overview (Technical report)*, February 2004. <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.

Möller, K., & Bojārs, U., & Breslin, J.G. (2006). Using Semantics to Enhance the Blogging Experience. In Y. Sure, J. Domingue (Eds.), *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006*, Budva, Montenegro, June 11-14, 2006, Proceedings, pp. 679–696.

Nelson, T.H. (1965). A File Structure for the Complex, the Changing, and the Indeterminate. In *ACM 20th National Conference Proceedings*, pages 84–100, Cleveland, Ohio, 1965.

Oren, E. (2006). An Overview of Information Management and Knowledge Work Studies: Lessons for the Semantic Desktop. In S. Decker, J. Park, L. Sauermaun, S. Auer, S. Handschuh (Eds.), *Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop (SemDesk 2006)* at ISWC 2006. Athens, GA, USA.

Quan, D., & Huynh, D., & Karger, D.R. (2003). Haystack: A Platform for Authoring End User Semantic Web Applications. In D. Fensel, K.P. Sycara, J. Mylopoulos (Eds.), *The Semantic Web – ISWC 2003: International Semantic Web Conference*, Proceedings, pp. 738–753.

Richter, J., & Völkel, M., & Haller, H. (2005). DeepaMehta – A Semantic Desktop. In *Proceedings of the 1st Workshop on the Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure* at ISWC 2005, Galway, Ireland.

Sauermann, L., & Grimnes, G. AA., & Kiesel, M., & Fluit, C., & Maus, H., & Heim, D., & Nadeem, D., & Horak, B., & Dengel, A. (2006). Semantic Desktop 2.0: The Gnowsis Experience. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo (Eds.), *The Semantic Web – ISWC 2006: 5th International Semantic Web Conference*, Athens, GA, Proceedings.

Sauermann, L. (2003). *The Gnowsis – Using Semantic Web Technologies to Build a Semantic Desktop*. Diploma Thesis, Technical University of Vienna, 2003.

Semy, S.K., & Pulvermacher, M.K., & Obrst, L.J. (2004). *Toward the Use of an Upper Ontology for U.S. Government and U.S. Military Domains: An Evaluation. (Technical report)*. MITRE, September 2004.

Sintek, M., & van Elst, L., & Scerri, S., & Handschuh, S. (2007). Distributed Knowledge Representation on the Social Semantic Desktop: Named Graphs, Views and Roles in NRL. In E. Franconi, M. Kifer, W. May (Eds.), *The Semantic Web – ESWC 2007: The 4th European Semantic Web Conference* (ESWC 2007), Proceedings.

Tummarello, T., & Morbidoni, C., & Nucci, M. (2006). Enabling Semantic Web Communities with DBin: An Overview. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo (Eds.), *The Semantic Web – ISWC 2006: 5th International Semantic Web Conference*, Athens, GA, Proceedings, pp. 943–950.

INDEXER REFERENCE LIST

Term 1 –mobile devices

- Also known as: "PDA", "Tablet Computer", "Smart Phone"
- Similar to: mobile technology
- Associated in the manuscript with: e-learning, m-learning, mobile learning
- Notable appearances of this term can be found on:
 - Page 3: context
 - Page 9: usage with learning objects
 - Page 19: accessibility